UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
INGENIERÍA TELEMÁTICA

PROYECTO FIN DE CARRERA

SERVICIO BITTORRENT PARA PLATAFORMAS DE SERVICIOS OSGi

Autora: Lydia Meléndez Roldán
Director: Mario Ibáñez Pérez
Año: 2010

AGRADECIMIENTOS

Aun me cuesta creer que estoy al final del túnel, ese túnel que con mis compañeros siempre comentábamos que nos costaba ver la luz, y a día de hoy la mayoría ya la estamos viendo.

El llegar aquí, ha sido gracias al apoyo de muchas personas importantes en mi vida.

Quiero agradecer el apoyo que he tenido siempre en mi familia, especialmente agradecer a mis padres, que siempre me dejaron total libertad en elegir mi camino, sin interponerse en ningún momento, y estando siempre apoyándome en los duros momentos. También me gustaría mencionar a mis abuelos, que aunque ya no estén aquí muchas veces son los que desde alguna parte me han ayudado a que se cumplan mis sueños.

A Lorenzo, porque si no hubiera estado a mi lado en todo este tiempo, creo que no hubiera pasado del primer cuatrimestre, dado que fue un poco desastroso y él fue quien me animó a seguir intentándolo.

A mis amigos, que han sufrido mis agobios, y que siempre me han definido como luchadora y me veían en el futuro como ingeniera. ¡Conseguido!

Otras personas muy importantes, mis compañeros de clase, especialmente Guio, mi compañera de prácticas durante todos estos años, con la que he pasado increíbles momentos y siempre nos hemos ayudado. También a Jorge, Crespo, Álex… por todos los buenos momentos vividos en clase y fuera de la Universidad.

Por último, quiero dar las gracias a Mario, porque este proyecto no habría tenido un final de no ser por su gran ayuda y apoyo. También, me gustaría destacar, que el trato recibido por su parte no podría mejorarse, porque ha sido maravilloso. Además, me ha ayudado a aprender y me ha demostrado, que aunque a veces las cosas no salen a pesar de probar una y otra vez, al final, de una manera u otra… ¡salen!

Gracias.

RESUMEN

En el presente proyecto, se ha realizado una aplicación que permite integrar un servicio de BitTorrent en plataformas de servicios OSGi.

BitTorrent ofrece la posibilidad de transferir archivos mediante una combinación de los protocolos P2P y FTP. Para ello disponemos de un servidor, al que llamamos *tracker*, que mediante el protocolo HTTP, nos permitirá la transferencia.

La integración consiste en la creación de dos servicios: uno de ellos permite la compartición y subida de archivos y otro permite su descarga.

Estos dos servicios, están basados única y exclusivamente en el protocolo BitTorrent. El servicio de compartición y subida, consiste en la creación de un archivo *.torrent*, que tendrá referenciado el archivo que se quiere compartir; este *.torrent* será anunciado y publicado en el servidor. El servicio de descarga, permite que un usuario a través de un cliente u otra aplicación, directamente, descargue archivos publicados en el servidor.

Para la interacción con la aplicación por parte del usuario final, se ha implementado un cliente que mediante comandos permite el uso de los servicios comentados.

ABSTRACT

This project presents an application to integrate a BitTorrent service in an OSGi service platform.

BitTorrent offers the possibility of sharing files combining two protocols, P2P and FTP. For this, we have the server, that we call "tracker", that allows with HTTP protocol, transfering of files .

The integration involves the creation of two services: one allows uploading and sharing, while the other allows downloading files.

These two services are only based on the BitTorrent protocol. The upload service has the functionality of create a file ".torrent" that refers to the file that is going to be shared. This ".torrent" will be announced and published on the "tracker". The download service allows a user through a client or other application, directly download files published in the server.

The end user can use the BitTorrent service by means of commands that have been implemented to allow the use of the services discussed.

ÍNDICE

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

ÍNDICE DE FIGURAS

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

ÍNDICE DE TABLAS

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

ACRÓNIMOS

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

- **API**: Application Programming Interface. Interfaz de programación de aplicaciones
- **JAR**: Archivo Java
- **OSGi**: Open Service Gateway Initiative
- **TCP/IP**: Transmission Control Protocol/Internet Protocol
- **P2P**: Peer to peer
- **FTP**: File Transfer Protocol
- **XML**: Extensible Markup Language
- **HTTP**: HyperText Transfer Protocol
- **LAN**: Local Area Network
- **VPN**: Virtual Private Network
- **JVM**: Java Virtual Machine
- **MIDP:** Mobile Information Device Profile
- **Java EE, SE, ME**: Enterprise Edition, Standard Edition, Micro Edition

1 INTRODUCCIÓN

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

1.1 *INTRODUCCIÓN*

En este proyecto confluyen 2 líneas principales de trabajo, de un lado los sistemas de transferencia de archivos P2P tan en boga, y de otro las plataformas de servicios que tienen su máxima representación en los entornos residenciales.

Desde hace unos años, la red Internet se ha ido imponiendo en nuestra sociedad. Esta red, nos da la oportunidad, a las personas, de mantenernos informadas en todo momento y desde cualquier lugar.

La aparición de Internet, se remonta a 1960 cuando el ARPA (Advanced Research Projects Agency) promueve un estudio sobre Redes cooperativas de computadoras de donde surgirá la primera red interconectada utilizando la red telefónica conmutada, ARPANET. Durante la década de los 70 se realizarán las primeras conexiones internacionales. Ya en 1982, se establecen el Protocolo de Control de Transmisión (TCP) y el Protocolo de Internet (IP) conocidos comúnmente como TCP/IP. Esto genera una de las primeras definiciones de Internet: "una serie de redes conectadas entre sí, específicamente aquellas que utilizan el protocolo TCP/IP".

La creación de las redes de ordenadores fue motivada principalmente para la compartición de archivos. En este sentido, en 1985 se completa el desarrollo de FTP (File Transfer Protocol), que nos permite la transmisión de archivos a través de Internet. Este protocolo irrumpió con fuerza siendo el más usado. Durante los últimos años han ganado interes las redes peer-to-peer (P2P) y programas para el fácil intercambio de archivos e información por parte de los usuarios como Ares, Emule o BitTorrent.

El principal objetivo de BitTorrent, es proporcionar de forma eficiente la distribución de un fichero para lo cual se basa en TCP y protocolos P2P. Se distribuye por medios convencionales un pequeño fichero con extensión .torrent. Este fichero es estático, por lo que a menudo se encuentra en páginas web o incluso se distribuye por correo electrónico. El fichero 'torrent' contiene la dirección de un servidor de búsqueda al que llamamos tracker, el cual se encarga de localizar posibles fuentes con el fichero o parte de él. El fichero será descargado de las fuentes que encuentra el tracker, el cual permite la transferencia mediante el protocolo HTTP. HTTP, es un protocolo de transferencia de hipertexto que define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

La gran aceptación de Internet, así como su gran evolución, ha dado lugar al desarrollo de nuevos dispositivos como la Pasarela Residencial, diseñada para conectar las infraestructuras de telecomunicaciones del hogar a redes públicas de datos como Internet.

La Pasarela Residencial normalmente combina las funciones de un router, de un hub, de un modem con acceso a Internet para varios PCs, de cortafuegos e incluso de servidor de aplicaciones de entretenimiento, como Vídeo/Audio bajo demanda, de comunicaciones, como VoIP (telefonía sobre Internet) o de telecontrol como la domótica.

La pasarela residencial, pretende servir de entrada en la sociedad digital multimedia y en el hogar, y con ello, acercar sus servicios al usuario mejorando su calidad de vida.La alianza Open Services Gateway Initiative (OSGi) fue creada en marzo de 1999 con el objetivo de diseñar una especificación software abierta, que permita construir plataformas compatibles que sean capaces de proporcionar múltiples servicios en el mercado residencial y automovilístico. Para ello, aprovecha las múltiples tecnologías que han ido apareciendo en el ámbito de los métodos de acceso como en el ámbito de la redes de datos y control de las viviendas o automóviles. Con todo ello, el OSGi pretende ofrecer una arquitectura completa y extremo-a-extremo, que cubra todas las necesidades del proveedor de servicios, del cliente y de cualquier dispositivo instalado en las viviendas **[4]**.

El proyecto realizado integra el programa de compartición de archivos BitTorrent en la comentada Pasarela Residencial OSGi. Para ello, se ha adaptado el protocolo, creando un servicio de subida y compartición de archivos en el servidor, y otro servicio de descarga de archivos. Para la parte de interacción con el protocolo, se ha creado una aplicación cliente, que a través de una serie de comandos, el usuario es capaz de realizar la tranferencia de archivo. En la figura 1, se muestra el entorno del protocolo BitTorrent sobre OSGi.

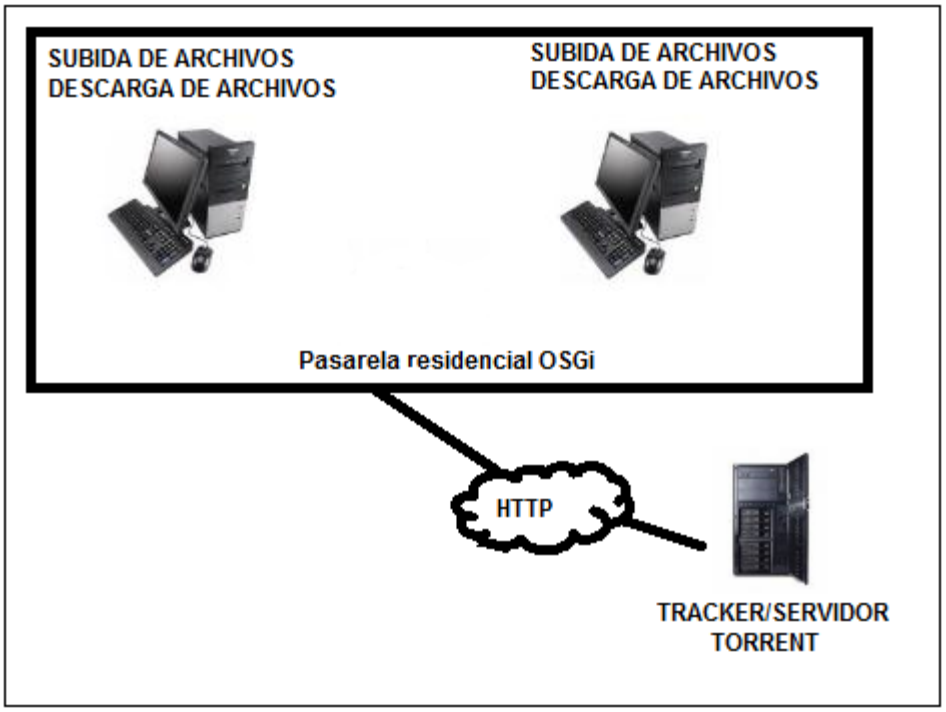

Figura 1: Entorno de BitTorrent sobre OSGi

1.2 *PLAN DE TRABAJO*

En primer lugar, será necesario comprender todo el entorno que va a rodear al protocolo BitTorrent, según la integración a realizar en el presente proyecto, así como realizar un estudio sobre la implementación de OSGi Apache-Felix..

Se estudiará profundamente el protocolo BitTorrent, cómo realiza la gestión de archivos tanto para subir como para descargar. Para la transferencia de archivos es necesario un tracker o servidor torrent que también habrá que estudiar a fondo.

Una vez estudiados estos dos puntos, se realizará una adaptación del código que proporcionado a través de la web oficial de BitTorrent para la posterior integración de dicha aplicación en una pasarela de servicios OSGi a través de la implementación de bundles.

Integrado el gestor BitTorrent en OSGi, se crearán otro bundle que hará las funciones de cliente cuya interacción con el usuario se hará mediante comandos.

1.3 *CONTENIDO DE LA MEMORIA*

A continuación se comentan las diferentes partes en las que se divide esta memoria:

CAPÍTULO 1 INTRODUCCIÓN: introducción acerca del presente proyecto, con la planificación prevista para su realización.

CAPÍTULO 2 OBJETIVOS Y MOTIVACIÓN: todos los objetivos necesarios para realizar integración de BitTorrent en OSGi, y que se han ido cumpliendo a lo largo de la realización del proyecto.

CAPÍTULO 3 ESTADO DEL ARTE: toda la tecnología empleada se describe en detalle.

CAPÍTULO 4 ESTUDIO DE VIABILIDAD: se estudia si el lanzamiento del proyecto sería rentable, y se incluye un presupuesto acerca de la cuantía económica, en función de las tareas realizadas y las horas empleadas en cada una de ellas.

CAPÍTULO 5 ANÁLISIS Y DISEÑO DEL SISTEMA: se analizan los requisitos que debe cumplir el proyecto.

CAPÍTULO 6 IMPLEMENTACIÓN: detalle de las decisiones tomadas a la hora de implementar la aplicación.

CAPÍTULO 7 PRUEBAS: fase en la que se prueba el sistema para asegurarnos del correcto funcionamiento.

CAPÍTULO 8 CONCLUSIONES: conclusiones sacadas al finalizar el proyecto.

CAPÍTULO 9 LÍNEAS FUTURAS DE DESARROLLO: descripción de las mejoras, posibilidades o ampliaciones de la aplicación a corto, medio y largo plazo.

BIBLIOGRAFÍA: detalle de todas las fuentes de información utilizadas en el proyecto.

APÉNDICE I MANUAL DE USUARIO: Guía de uso orientada a nuevos usuarios

APÉNDICE II DESCRIPCIÓN DEL CD: CD que contiene la aplicación, el manual del programador, los programas utilizados y la documentación necesaria.

2 OBJETIVOS Y MOTIVACIÓN

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

El objetivo general del presente proyecto es desarrollar un servicio para plataformas de servicios OSGi. Este servicio permitirá compartir ficheros mediante el protocolo P2P funcionando como clientes de otros servicios de la plataforma.

El proyecto deberá completarse con la creación de una aplicación que funcione dentro de la plataforma y que demuestre la funcionalidad proporcionada por dicho servicio de BitTorrent.

Para conseguir todas las funcionalidades, se diseñarán diferentes servicios de manera que BitTorrent sea compatible con OSGi. Para ello, se distribuirá la aplicación en bundles que se interrelacionarán entre sí.

OBJETIVO 1: Integración del protocolo BitTorrent en la plataforma de servicios OSGi.

OBJETIVO 2: Creación de un servicio que permita subir y compartir archivos en un servidor. Este servicio permitirá la creación del archivo con extensión .torrent, anunciar este archivo en el tracker, publicarlo y compartirlo.

OBJETIVO 3: Creación de un servicio que permita la descarga de archivos. Este servicio, permitirá a un usuario, descargarse el archivo publicado en el tracker.

OBJETIVO 4: Creación de un bundle que use los servicios de los objetivos 2 y 3. Se implementarán unos comandos necesarios para la interacción por parte del usuario. A través de estos comandos, podremos crear el .torrent, publicarlo, compartirlo y descargarlo.

OBJETIVO 5: Integración del tracker en la plataforma de servicios OSGi, dando la posibilidad de que el servidor se pueda encontrar en la pasarela residencial.

Completados estos objetivos, se hará una demostración del funcionamiento de la aplicación. Un cliente publicará un archivo en el servidor, y una vez compartido, otro cliente tendrá que ser capaz de descargárselo.

Para la parte de comprobación, podrán hacerse diferentes pruebas, como compartir varios archivos, y que solo un cliente se descargue todos, o compartir varios archivos y que diferentes clientes se descarguen varios archivos. En definitiva, la aplicación tendrá todas las funcionalidades de uso que posee BitTorrent sin ser integrado en la plataforma OSGi.

3 ESTADO DEL ARTE

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

En esta sección de estado del arte, se procederá a explicar el protocolo Bittorrent, y puesto que va a ser integrado en una pasarela residencial se verá en qué consisten las pasarelas residenciales, en especial OSGi, así como una introducción a la domótica. Además se analizarán las tecnologías utilizadas en la realización del proyecto.

3.1 *PASARELA RESIDENCIAL*

Una Pasarela Residencial es un dispositivo que conecta las infraestructuras de telecomunicaciones (datos, control, automatización, ...) del hogar digital a una red pública de datos, como por ejemplo Internet. La Pasarela Residencial normalmente combina las funciones de un router, de un hub, de un modem con acceso a Internet para varios PCs, de cortafuegos e incluso de servidor de aplicaciones de entretenimiento, como Vídeo/Audio bajo demanda, de comunicaciones, como VoIP (telefonía sobre Internet) o de telecontrol como la domótica.

Permite la conectividad total de los hogares con el mundo exterior para poder telecontrolar electrodomésticos, sistemas de seguridad, de domótica, de gestión energética, equipos de electrónica de consumo como vídeos y TVs, ordenadores personales y muchos más. En la figura 2, se muestra un ejemplo de necesidad de pasarela residencial.

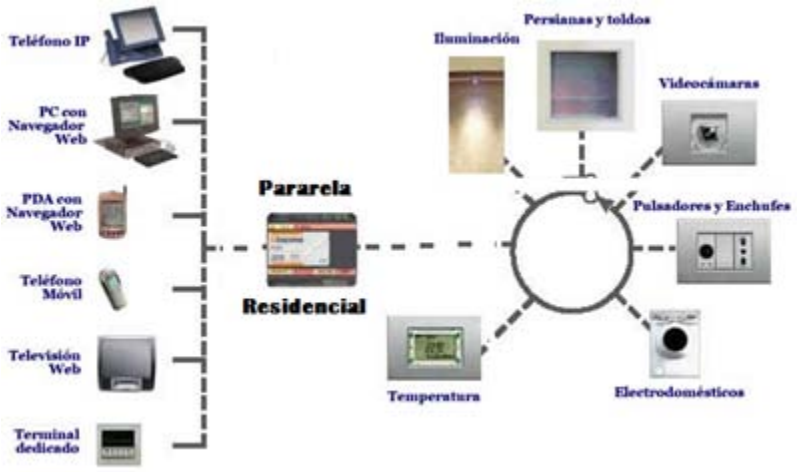

Figura 2: Pasarela Residencial

La domótica, ayuda a explicar esta necesidad de pasarela residencial, por lo que a continuación se presenta una breve introducción a ella.

Definimos domótica como la automatización del hogar o vivienda, ya esté en una casa aislada o en un piso de un inmueble, y sea cual sea su grado y las tecnologías empleadas.

No debemos confundir la automatización de edificios tales como oficinas, despachos y servicios en general, puesto que este tipo de automatización es conocida como inmótica.

En definitiva, el uso de las técnicas y tecnologías disponibles (electricidad, electrónica, informática, robótica, telecomunicaciones…), logra una mejor utilización, gestión y control de todos los aspectos relacionados con la vivienda (confort, seguridad, ahorro de consumo de energía, comunicaciones, informática, televisión, cine en casa....) [6].

En la figura 3, se muestra un ejemplo de una vivienda con la incorporación de domótica.

Figura 3: Vivienda con la incorporación domótica

La domótica aporta numerosos beneficios, que crecen y se actualizan a diario. Es capaz de proporcionar a la vivienda tradicional, la posibilidad de controlar y gestionar de forma eficiente los sistemas existentes y equipos ya instalados (Sistemas de alarma, TV, teléfono, agua, cocina, refrigerador, eléctrico,…), mediante un sistema de gestión técnica inteligente, con el objetivo de permitir una mejor calidad de vida al usuario de dicha vivienda [5]. Las principales áreas socio-técnicas y sus funciones que incluyen la domótica son:

- *Automatización y control*: Abrir, cerrar, apagar, encender, regular… Dispositivos y actividades domésticas (iluminación, climatización, persianas, toldos, puertas, ventanas, cerraduras, riego, electrodomésticos, suministro de agua, gas, electricidad….).
- *Gestión energética*: Conexión de dispositivos de calefacción y aire acondicionado según criterios de ahorro y confort, complemento de control de toldos y persianas para aprovechamiento de las energías naturales, control de alumbrados, racionalización de cargas eléctricas.
- *Seguridad*: Vigilancia automática de personas, incidencias y averías, alarmas de intrusión y cámaras de vigilancia, alarmas personales, alarmas técnicas de incendio, humo, agua, gas, fallo de suministro eléctrico… Además, la domótica facilita la introducción de infraestructuras y la creación de escenarios que complementan las siguientes áreas socio-técnicas provenientes de los nuevos avances en la Sociedad de la Información.
- *Comunicaciones*: Transmisión de voz y datos, incluyendo textos, imágenes, sonidos (multimedia) con redes locales (LAN) compartiendo acceso a Internet, recursos e intercambio entre todos los dispositivos, acceso a nuevos servicios de telefonía sobre IP, televisión digital, televisión por cable, diagnóstico remoto, videoconferencias…
- *Mantenimiento*: Tiene la capacidad de incorporar telemantenimiento de los equipos.
- *Ocio y Tiempo Libre*: Descansar y divertirse con radio, televisión, multi-room, cine en casa, videojuegos, captura, tratamiento y distribución de imágenes fijas y dinámicas y de sonido (música) dentro y fuera de la casa, a través de Internet.
- *Salud*: Actuar en la sanidad mediante asistencia sanitaria, consultoría sobre alimentación y dieta, telecontrol y alarmas de salud, medicina monitorizada, cuidado médico…
- *Compra*: Comprar y vender mediante la telecompra, televenta, telereserva, y todo ello desde casa.
- *Finanzas*: Gestión del dinero y las cuentas bancarias mediante la telebanca, consultoría financiera.
- *Aprendizaje*: Aprender y reciclarse mediante la tele-enseñanza, cursos a distancia y otras actividades.
- *Actividad profesional*: Trabajar total o parcialmente desde el hogar, posibilidad viable para ciertas profesiones (teletrabajo) y ciertos perfiles psicológicos.

- *Ciudadanía*: Gestiones múltiples con la Administración del Estado, la Comunidad Autónoma y el Municipio. Un ejemplo es el voto electrónico.
- *Lecturas*: Búsqueda y procesamiento de otra información: Museos, bibliotecas, libros, periódicos, información meteorológica, jurídica, fiscal…
- *Otros*: Todas las posibles ideas que la creatividad y la innovación puedan aportar. Lo comentado en estos apartados es sólo una muestra del actual estado de conocimiento y progreso.
- Otras ventajas de la domótica para el usuario son:
    - Un hogar más seguro: se consigue a través del sistema de seguridad comentado anteriormente.
    - Un hogar más confortable: a través del control de electrodomésticos, climatización, persianas motorizadas, control remoto de equipos e instalaciones…
    - Mejor comunicación: evitar el aislamiento de las personas, un ejemplo claro es la teleasistencia.

Mayor sostenibilidad: aprovechamiento máximo de energía, luz solar, control de consumo y en definitiva un hogar menos contaminante.

3.1.1 EVOLUCIÓN

--- --- --- --- --- --- --- ---

Hace 8 años, una pasarela era un simple modem, que a día de hoy a evolucionado hasta convertirse en todo un computador capaz de gestionar múltiples servicios. En la figura 5, podemos ver eje cronológico de la evolución.

Por un lado se ha pasado de dar conectividad mediante ADSL a evoluciones de esta tecnología o algunas nuevas como la fibra óptica o las comunicaciones sin cables. También se ha producido evolución tanto en la conectividad de los aparatos como en como estos podían comunicarse con la pasarela, siendo en un principio un único PC y llegando a una red completa donde casi cualquier electrodoméstico del hogar puede estar conectado.

Esta evolución se ha producido en parte por una evolución de los servicios disponibles a través de Internet que primero fueron de datos, después de voz con la aparición de herramientas de Voz sobre IP y finalmente con la llegada del video con la TV sobre IP. Este aumento de servicios a provocado que ya no hablemos de redes de datos sino del “Triple Play”

don de las redes proporcionan datos, audio y video y que estén evolucionando hacia el "Multiplay" donde no habrá diferenciación de estos tipos de datos.

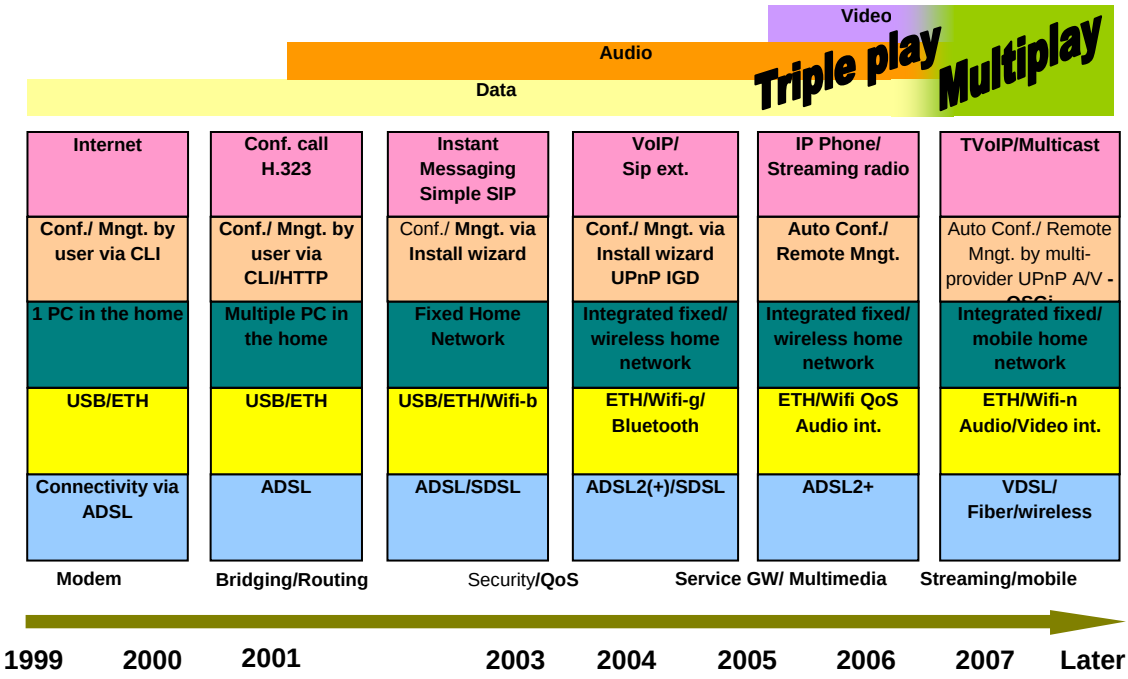

Figura 4: Evolución de Pasarela Residencial [28]

Hay todavía un factor más que ha evolucionado junto con la pasarela y este es el modo de configuración y gestión. Inicialmente uno disponía de sencillas aplicaciones de usuario capaces de modificar ciertos parámetros del entonces modem o router. Al hacerse más compleja la configuración se hizo necesaria mejorar estas aplicaciones con "wizards" que ayudaban al usuario en este proceso o dispositivos que utilizan protocolos de configuración automática. El siguiente paso supone un avance tanto en la transparencia hacia el usuario como en el aumento del control por parte del proveedor sobre la pasarela ya que incluye la posibilidad de la configuración y gestión remota de dicha pasarela.

El siguiente paso en el campo de la configuración y gestión de dispositivos es de proporcionar a la pasarela de un mecanismo que proporcione, no solo una configuración y gestión remotas y automáticas sino también que permita a la pasarela adaptarse modificando su comportamiento, esto es, que la pasarela pueda proporcionar por si misma nuevos servicios según se demanden. Además la pasarela debe afrontar el reto de tener a diferentes proveedores tratando de gestionarla.

3.1.2 APLICACIONES

Dadas las numerosas aplicaciones de pasarela residencial, quizás la que más interés presenta a corto plazo es la compartir, de forma simultánea, el acceso a Internet entre varios PCs o equipos de entretenimiento de la vivienda.

Las aplicaciones no están limitadas por el acceso de banda ancha a Internet sino que, gracias a la aparición de nuevos operadores y proveedores, surgirán nuevos servicios de valor añadido (e-services) más útiles que el simple acceso a Internet [28], destacan:

- *Comunicaciones*: e-mail, acceso compartido a Internet, Voz sobre IP (VoIP), cortafuegos (firewall), gestión de direcciones IP y otras.
- *Telecontrol* y *Telemetría*: con aplicaciones domóticas al frente. Destacan la telegestión energética, el control remoto de electrodomésticos y equipos, el diagnóstico de los mismos y el uso de Webcams que permitan observar lo que está ocurriendo en ciertas zonas o habitaciones de la vivienda.
- *Seguridad*: custodia y vigilancia de hogares e instalaciones, alarmas de intrusión, de incendio, médicas etc.
- *e-commerce*: venta de productos y servicios usando la pasarela como método de acceso y, por lo tanto, escaparate de los mismos, además de proporcionar autenticación de los usuarios y interfaces para métodos de pago con smartcards.
- *Entretenimiento*: puede servir como plataforma para Vídeo/Audio bajo demanda, juegos en red, charlas (chat rooms), etc.

3.1.3 CARACTERÍSTICAS

Los servicios tienen que aportar valor, confort y tranquilidad en lo modo de vida para que su implantación tenga éxito y realmente el equipo pueda ser catalogado como Pasarela Residencial. Se han definido unas determinados características que deben tener todas las Pasarelas Residenciales. A continuación se detallan los requisitos:

- La instalación debe ser sencilla y la configuración rápida y asequible (mejor si es Plug&Play). Una vez conectada a la roseta telefónica o a alguna de las bocas del modem de cable o del router ADSL, la configuración debe ser inmediata. Igualmente, la asignación y especificación de las funciones que puede hacer cada dispositivo domótico o electrodoméstico debería ser automática.

- Telecarga de software. El proveedor de servicios, o directamente el usuario, bajo supervisión del proveedor, deberían ser capaces de actualizar o telecargar nuevos servicios, además de configurarlos remotamente.
- Soporte para redes. Las Pasarelas Residenciales deberían tener interfaces que permitan conectar redes de datos de banda ancha (>10Mbps) con tecnologías como la tradicional Ethernet o con las nuevas tecnologías "sin cables" como HomePlug, HomePNA, HomeRF o 802.11b.
- Por otro lado sería interesante que tuvieran interfaces para redes de control de banda estrecha (red domótica) que permitan implementar funciones de telecontrol y ahorro energético.
- Cortafuegos y capacidad de construir VPNs. Deben ofrecer servicios de protección de los datos y seguridad contra los ataques de los hackers, impidiendo el acceso de estos a los ordenadores o equipos en red de la vivienda (cortafuegos). Deberían ser capaces de formar redes privadas virtuales entre estos equipos (Virtual Private Network, VPN).
- Capacidad para soportar múltiples servicios. Con suficiente memoria, capacidad de procesamiento y un sistema operativo embarcado robusto y multitarea, las pasarelas residenciales deberán ser capaces de ejecutar múltiples aplicaciones concurrentemente, donde cada una de ellas se corresponderá a un e-service diferente. La conexión de banda ancha será compartida entre todos estos servicios con la multiplexación de datos, ya sea a nivel IP o nivel de aplicaciones.
- Monitorización usando páginas Web. Ya sea de forma local, como de forma remota, el usuario podrá acceder a las tripas de la Pasarela Residencial para cambiar su configuración, borrar aplicaciones (servicios) o supervisar su estado. Para ello las pasarelas tendrán que tener embarcados pequeños servidores HTTP o WAP.

3.1.4 TIPOS DE PASARELAS RESIDENCIALES

--- --- --- --- --- --- --- ---

Equipos con prestaciones dispares son considerados pasarelas por lo que hay una limitación en los tipos. Se vienen clasificando en dos tipos:

- *Pasarelas Residenciales de Banda Ancha*: son routers/hubs o modems ADSL o de Cable que actúan como pasarelas en sí mismas, adaptando entre los datos de la red interna de la vivienda y la conexión de banda ancha de Internet. Suelen tener interfaces para cable Ethernet categoría 5 o bocas USB, aunque ahora ya empiezan a aparecer modelos con acceso inalámbrico con 802.11b o aprovechando la instalación telefónica

de la vivienda (HomePNA). Este tipo de pasarelas está en auge gracias a el aumento del teletrabajo y las pequeñas oficinas de profesionales liberales (Small Office/Home Office, SOHO). En el sentido estricto no se pueden considerar a este tipo de equipos como una Pasarela Residencial, pero si bien es cierto, cada vez proporcionan más funciones y servicios totalmente personalizables de cara al usuario según el proveedor u operador de conexión a la banda ancha. Por lo tanto, teniendo en cuenta que se pueden personalizar para ofrecer los servicios demandados por el usuario, se puede decir que este tipo de equipos son una primera generación de Pasarelas Residenciales.

- *Pasarelas Residenciales Multiservicios*: proporcionan varios interfaces para redes de datos y control con diferentes tecnologías, además de ser más complejas y potentes. Son capaces de ejecutar diferentes aplicaciones (servicios) con requisitos de tiempo real (para VoIP o streaming de vídeo para Pay-per-View). También puede ejecutar servicios orientados a las SOHOs como el acceso único a Internet para varios PCs.

3.1.5 IMPLANTACIÓN

--- --- --- --- --- --- --- ---

Hay que destacar que la funcionalidad de una pasarela residencial puede ser implementada de diversas formas. Basta con un simple PC de sobremesa, algunas tarjetas específicas y una aplicación SW construida para tal fin. Hay funciones básicas de las pasarelas que pueden ser implementadas con un pequeño software embarcado dentro de las set-top boxes de TV por cable o por satélite, de las consolas de videojuegos, de las centralitas telefónicas, de los routers ADSL.

El tipo de Pasarela más implementado es la Multiservicios. El aspecto de una pasarela de este tipo es el de una caja cerrada, sin interfaces visuales, ni teclados ni discos duros. Está construida con electrónica probada y fiable, y no necesita administración por parte del usuario.

También se podrán encontrar pasarelas residenciales insertadas dentro del frigorífico y con una pantalla en la puerta de este. En esta residen la mayoría de los electrodomésticos susceptibles de telecontrolarse. Además un frigorífico es un electrodoméstico que siempre está encendido y que tiene un coste y un tamaño que lo hacen ideal para empotrar los circuitos de una potente pasarela.

3.1.6 INTERFACES

Las Pasarelas Residenciales tendrán interfaces que permitirán intercambiar información con cualquier equipo, dispositivo o electrodoméstico que tenga conectividad para redes de datos o de control. Algunos de estos equipos son:

- Ordenadores de sobremesa y portátiles
- Reproductores MP3 (portátiles o fijos dentro del equipo HiFi), sintonizadores de emisoras de radio Internet.
- DVDs y Televisores
- Pizarras portátiles o Web Pads Agendas personales o PDAs
- Videoconsolas con juegos en red
- Teléfonos móviles
- Electrodomésticos
- Equipos de supervisión médica y alarmas de pánico
- Centralitas de custodia y alarmas técnicas
- Instalaciones domóticas y los contadores de luz, agua y gas

Las posibilidades son infinitas, y todas las tecnologías de interconexión para estas interfaces están disponibles. La figura 4 es un ejemplo de interfaz.

Figura 5: Interfaz de un teclado domótico

3.2 *OSGi*

OSGI son las siglas de Open Services Gateway Initiative.Su objetivo es definir las especificaciones abiertas de software que permita diseñar plataformas compatibles que puedan proporcionar múltiples servicios. Fue pensado principalmente para su aplicación en redes domésticas y por ende en la llamada Domótica o informatización del hogar.

OSGi Alliance se fundó en 1999, alianza Java basada en una plataforma de servicios que pueden ser gestionados de forma remota. Se crea como marco principal con un ciclo de vida, un registro de servicio, un entorno de ejecución y los módulos.

La especificación de OSGi se ha definido con una serie de APIs básicas para el desarrollo de servicios, como los de logging, servidor HTTP y el Device Access Specification o DAS, que permite el descubrir los dispositivos y servicios ofrecidos por éstos.

La arquitectura de OSGi posee dos elementos fundamentales de los cuales el Service Platform está situado en la red local y conectada al proveedor de servicios a través de una pasarela en la red del operador. Este elemento será el responsable de permitir la interacción entre dispositivos o redes de dispositivos que podrían utilizar distintas tecnologías para comunicarse.

3.2.1 ARQUITECTURA OSGi

OSGi proporciona un entorno de computación para Bundles que se ejecutan conjuntamente en una JVM.Permite a las aplicaciones compartir una única JVM.

En cuanto a la carga de clases las gestiona de una manera mejor definida y eficiente que el estándar Java, soportando incluso algunas versiones. Entre aplicaciones, provee aislamiento y seguridad, pero permitiendo comunicación y colaboración. Provee la gestión del ciclo de vida, cuyas fases son Instalar, Empezar, Parar y Actualizar, Es una arquitectura libre de políticas impuestas únicamente por los bundles, como se muestra en la figura 6.

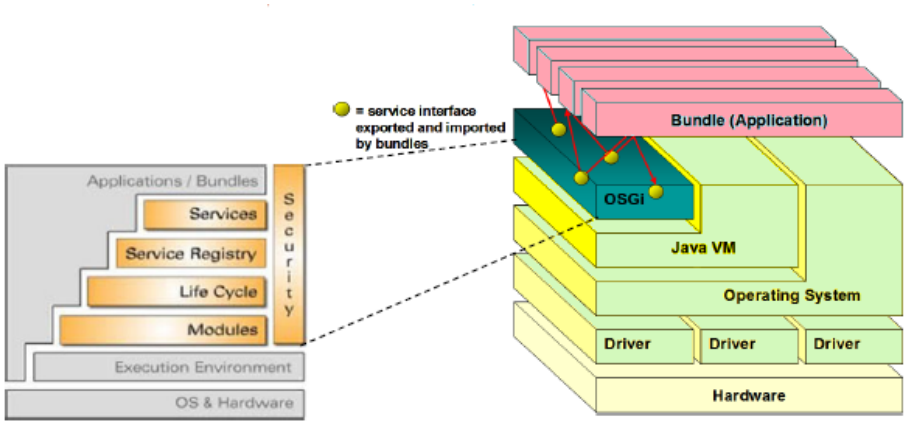

Figura 6: Arquitectura OSGi

Es una arquitectura orienta a objetos. Separa la definición de interfaz de la implementación permitiendo implementaciones alternativas. Las aplicaciones son asociadas dinámicamente.

Es posible la reutilización de componentes, y estos componentes pueden ser desligados a los detalles de implementación de otros componentes, únicamente tienen que conocerse sus interfaces.

OSGi pretende crear una plataforma que sea capaz de procesar y tratar de forma correcta toda la información necesaria para proporcionar servicios de comunicaciones, de entretenimiento, de telecontrol o teledomótica, y de seguridad. Por lo tanto, la especificación OSGI debe tener los interfaces adecuados para soportar todos estos servicios sin incompatibilidades además de permitir gestionarlos de forma adecuada. La figura 7, muestra un marco de aplicación de OSGi.

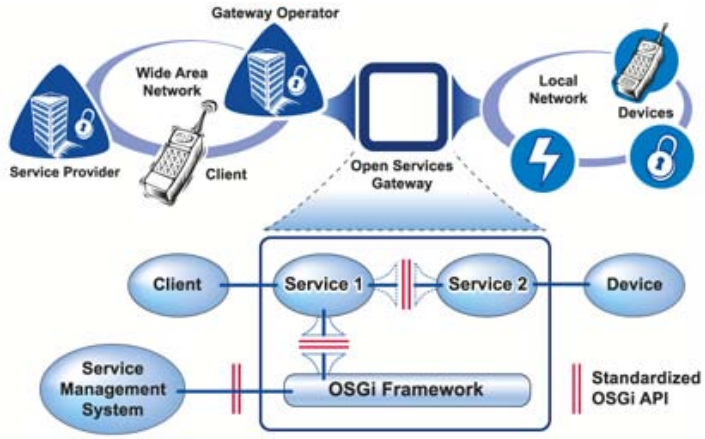

Figura 7: Marco de aplicación OSGi

3.2.2 BUNDLES

--- --- --- --- --- --- --- ---

Definición e infraestructura.

Una sencilla definición de bundle, podría ser una aplicación ejecutable contenida en varias implementaciones OSGi. Lo que hace el bundle es registrar servicios, o lo que es lo mismo, especifica interfaces. La especificación OSGi tiene definidos unos servicios estándar (Bundle repository) que las aplicaciones pueden cargar y descargar.

Los bundles son relacionados a través de objetos de servicios y de la compartición de paquetes. Un registro de servicios dinámico permite a un bundle encontrar y seguir el rastro de otros objetos de servicio. Como ya se ha comentado anteriormente hay una buena seguridad y dependencias en lo referente a gestión por parte del framework.

Las dependencias pueden ser creadas en un bundle completo(Required-Bundle) o en un único paquete(Import-Package), y en este último caso los bundles solo importarían lo que necesitan, sin embargo, en el primer caso se importarían paquetes que no van a tener por qué ser utilizados.

Contenido Interno.

Los metadatos del bundle están contenidos en el fichero manifest.mf y un bundle es un fichero con extensión .jar. Este jar contiene todo el código java necesario para la aplicación.

Cuando procedemos a instalar un bundle lo que sucede es que el manifest empieza a ser leído e instala el código y recursos (otros ficheros dentro del .jar) resolviendo las dependencias e iniciando así el control del ciclo de vida del bundle [1].

Una vez instalado, procedemos a la ejecución invocando al BundleActivator, que contiene los métodos start y stop(como ya se ha comentado en el ciclo de vida), que elimina los recursos utilizados por el bundle cuando acaba.

Para el funcionamiento del bundle, deberán estar disponibles en la maquina una serie de servicios o paquetes.

La dependencia a otros recursos, como los paquetes JAVA, deben estar disponibles para que el bundle pueda trabajar que las dependencias a estos paquetes se resuelven antes de arrancar el bundle.

Las cabeceras pre-establecidas más importantes del manifest, para especificar los parámetros que son necesarios para instalar y activar un bundle son [3]:

- Manifest-Version: Indica la versión del manifiesto.
- Created-By: Indica al autor del *bundle*.
- Bundle-Name: Indica el nombre del *bundle*.
- Bundle-Description: Contiene una breve descripción del *bundle*.
- Bundle-Vendor: Distribuidor del *bundle*.
- Bundle-Copyright.
- Bundle-Vension: Vensión del *bundle*.

- Bundle-Activator: Indica cual es la clase principal del *bundle* la que contiene el método start.
- BundleClassPath: Directorio en el cual se buscan los ficheros jar.
- Import-Package: Indica cuáles son los paquetes o servicios necesarios para poder ejecutar el *bundle*.
- Export-Package: Servicios que se exportan para ser utilizados para otros bundles.

En la figura 8, se ve como es el empaquetamiento de un bundle. Un archivo .jar que al descomprimirlo tenemos dentro los paquetes con las clases y el MANIFEST.

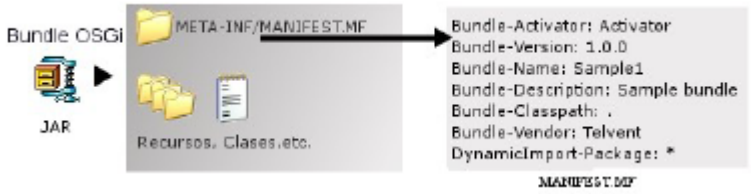

Figura 8: Empaquetamiento de Bundles en OSGi

Gestión del ciclo de vida de bundles

Desde que se instala un bundle, éste va variando de estado hasta que se desinstala. Todos esos estados constituyen el ciclo de vida de un bundle, y vienen definidos en la especificación de OSGi. En resumen serían los siguientes pasos:

1. Instalar un *bundle*
2. Arrancar/Parar un *bundle*
3. Actualizar un *bundle*
4. Desinstalar un *bundle*

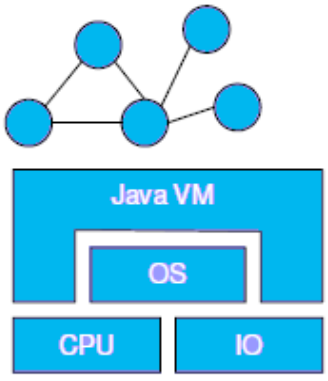

Figura 9: Componentes software, ciclo de vida

En el apartado framework de OSGi, una de las capas se corresponde con el ciclo de vida del bundle y ahí veremos en detalle los estados por los que pasa.

OSGi está preparado para dispositivos que operan controlados por un operador de plataforma que requiera gestión remota, por lo que el ciclo de vida no para cuando un dispositivo abandona, y es conveniente actualizar el software instalado una vez desplegado. Para permitir gestión remota,

OSGi proporciona una API de gestión de bundles, donde algunos bundles autorizados actúan de puente entre cualquier protocolo y las llamadas del API. Es el único modelo de servidor de aplicaciones Java donde éstas pueden compartir código entre ellas y no se ejecutan aisladas unas de otras, a diferencia de MIDP o Java EE. Aporta un modelo de servicios ligero que permite publicar, encontrar y asociar servicios dentro de una JVM a través del registro de servicios.

Proceso de despliegue de los bundles:

Cuando se crea una aplicación se debe ser cuidadoso con lo que se asocia a esta. La instalación de las dependencias es suficiente para el funcionamiento, pero no debemos asociar a la aplicación solamente las nuevas dependencias instaladas, sino también las que ya estaban instaladas en la plataforma.

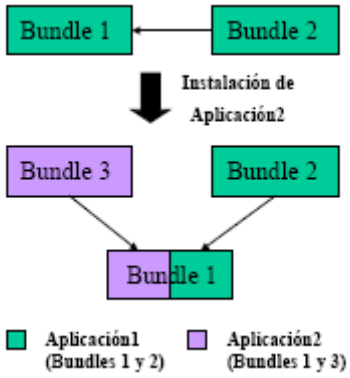

Figura 10: Bundles y Aplicaciones

Según la figura 10, la Aplicación1 depende de los bundles Bundle1 y Bundle2 y la Aplicación2 depende de los bundles Bundle1 y Bundle3. [24]

Si instalamos Aplicación1 se instalarán los bundles 1 y 2 pues son nuevos en la plataforma. Al instalar Aplicación2, si sólo asociásemos los nuevos bundles instalados estaría compuesta únicamente por Bundle3, se podría crear una inconsistencia en el futuro si se para o desinstala Aplicación1. Para evitarlo hay que asociar a la aplicación no sólo los nuevos recursos instalados sino todos los que se encuentran en el árbol de dependencias resultado del análisis. De esta forma, al instalar Aplicación2 el resultado del análisis de las dependencias debe dar como

resultado que se asocien los bundles 1 y 3 a esta aplicación. En este caso, parar o desinstalar Aplicación1 no afecta a Aplicación2.

3.2.3 FRAMEWORK OSGi

--- --- --- --- --- --- --- ---

- **Entorno de ejecución seguro:**

El modelo de seguridad ofrecido por OSGi, se divide en 4 niveles:

1 Un mecanismo de seguridad de la JVM que previene operaciones peligrosas como manipulación de punteros o acceso no restringido a arrays.

2 Seguridad del lenguaje Java a través de los modificadores de acceso: public, prIvate, protected.

3 Seguridad proporcionada por la plataforma Jave SE (java.security.permission)

4 OSGi separa unos *bundles* de otros y comprueba que un *bundle* tiene permisos para interactuar con otro.

- **CAPAS DE LA Framework OSGi:** Como se indican en la figura 11, son las siguientes

(1)*ENTORNO DE EJECUCIÓN*

(2)MÓDULOS

(3)CICLO DE VIDA(Bundles)

(4)SERVCIOS

(5)SEGURIDAD

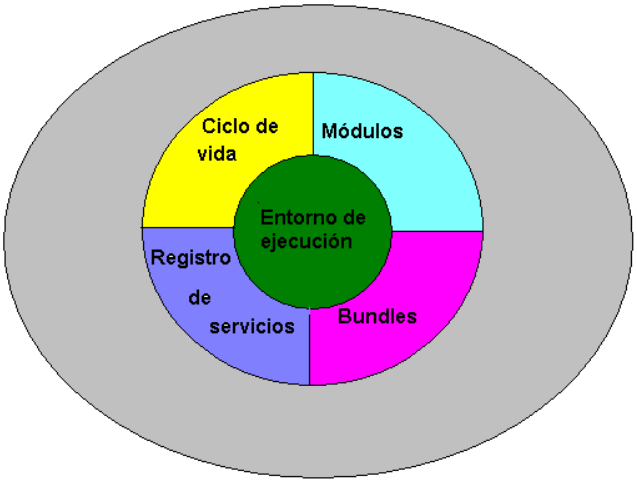

Figura 11: Capas del framework de OSGi

(1)Entorno de ejecución

Requiere un entorno de computación seguro y robusto. Este entorno está inspirado en Java. Las clases OSGi, utilizan un subconjunto de las clases definidas por Java SE o Java ME CDC/CLDC (ver figura 12).

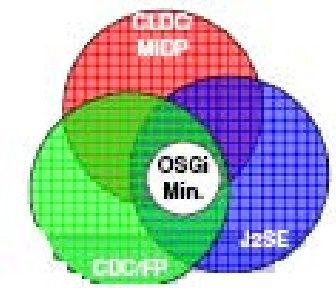

Figura 12: Entorno de ejecución

(2)Modularidad

Un bundle es la unidad de ejecución en OSGi, y está compuesto por clases de Java, librerías, manifest. Los bundles eliminan dependencias en classpath y protegen quien puede acceder a que información. Además como ya he comentado anteriormente puede soportar distintas versiones.

OSGi promueve compartir las clases entre bundles, ya que un bundle puede utilizar librerías utilizadas por otros bundles, lo que suponen un ahorro de memoria.

Cada bundle puede exportar e importar paquetes (conjuntos de clases).Si múltiples bundles exportan el mismo paquete (con una versión diferente), el framework ha de seleccionar una versión apropiada por cada bundle. Tras desinstalar un bundle los importadores son reiniciados para que se asocien a otro nuevo exportador de paquetes. Las dependencias sobre otros bundles puede ser con bundles todavía no instalados.

(3)Ciclo de vida. Bundles

El framework de OSGi ofrece un API paralau gestión de bundles.

Pasos del ciclo de vida de un *bundle*:

-Instalación

-Resolución

-Arranque

-Parada

-Refresco

-Actualización

-Desinstalación

Cuando instalamos un bundle será necesario resolver sus dependencias para poder arrancarlo. Todo bundle debe implementar la interfaz BundleActivator con los métodos start y stop. La cabecera del manifest.mf (del cual se habla un poco más adelante), indicará la clase que hay que instanciar para arrancar o parar un bundle. Se tiene en cuenta que en el ciclo de vida de un bundle pueden ocurrir ciertos eventos en el framework como la actualización o instalación de otros bundles.

Si un bundle con dependencias a otro es desinstalado, para acceder tendrá que acceder a los paquetes exportados y hacer un refresco del bundle. El comienzo y final de los bundles son grabados permanentemente, y a través de los start-levels nos indican el orden de arranque de los bundles en el framework.En la figura 13 se observan todos los posibles estados del bundle.

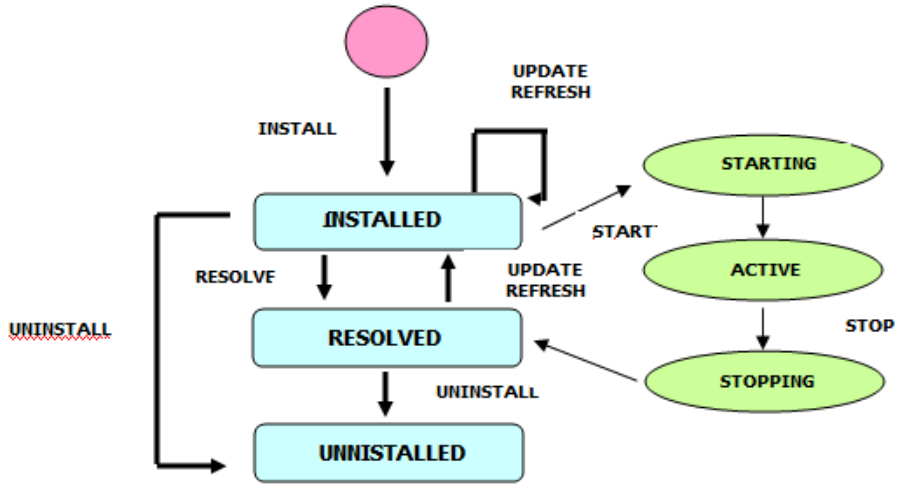

Figura 13: Posibles estados de un bunlde

(4)Servicios

La plataforma OSGi, provee un modelo de servicios dentro de la JVM y define un gran conjunto de servicios. Se basa en la seguridad de Java2, es decir, en permisos y firma de bundles.

Un servicio es un interfaz java con propiedades dependiendo del tipo de servicio. Los servicios son implementados como objetos dentro del bundle. La plataforma OSGi está orientada a servicios del tipo:

-Proveedores: publicación de descripciones del servicio.

-Consumidores: descubren y usan el servicio.

-Registro: los consumidores pueden descubrir otros servicios a través de consultas

El registro de servicios permite el diseño de componentes que pueden ser adaptados a un entorno cambiante. Además, estos componentes pueden ser combinados para crear aplicaciones complejas. La principal diferencia con otros entornos es su naturaleza dinámica en lo que a dispositivos electrónicos se refiere.

Un servicio no deja de ser un objeto java dentro de un bundle. La semántica del servicio viene dada por su interfaz Java, pudiendo registrar un bundle uno o más servicios.

La interfaz BundleContext facilita los métodos para manipular el registro de servicios. Los registros de servicio son gestionados por objetos de tipo ServiceRegistration y pueden utilizarse para desregistrar un servicio o modificar sus propiedades. Además, estos objetos, ServiceReference dan acceso al servicio así como a sus propiedades. Este acceso se realiza mediandte el getService y se devuelve con el ungetService.

(5)Seguridad

Esta plataforma permite la reutilización de componentes para diferentes aplicaciones, y tiene una gran aceptación en el mercado. También, presenta un modelo dinámico para la personalización y variación de aplicaciones en los dispositivos de hoy en día, con un estricto control del sistema de gestión. El acceso a los recursos del sistema está protegido a través de Código Java 2 (java.security.Permission) y sus subclases (FilePermission y SocketPermission).

En el caso de querer exponer el contenido de un bundle, se hará mediante los modificadores de acceso de las clases, haciendo que ciertos paquetes sean visibles. La importación o exportación de paquetes está regulada por OSGi.

Los servicios pueden ser divididos en dos grupos, servicios del framework y servicios del sistema:

- Servicios del framework
  - *Permission Admin:* los permisos de *bundles* actuales o futuros pueden ser manipulados a través de este servicio.
  - *Conditional Permission Admin:* extiende el Permission Admin con permisos que son aplicados cuando ciertas condiciones son cumplidas.
  - *Package Admin:* provee información sobre el estado compartido por un paquete y permite refrescarlos.
  - *Start Level:* conjunto de *bundles* que deben ejecutarse conjuntamente precediendo o siguiendo a otros niveles de comienzo.

- *URL Handlers: permite a los* bundles *contribuir dinámicamente.*

- Servicios del sistema **[25]**

  Log Service: trazado de información, advertencias, información de debug o errores es redireccionado a bundles subscritos con él.

  - *Configuration Admin: modelo flexible y dinámico para obtener información de configuración.*
  - *Event Admin: mecanismo general y flexible de publicar y subscribirse a eventos.*
  - *Device Access: mecanismo para asociar un driver a un nuevo dispositivo y descargar automáticamente un bundle.*
  - *User Admin: BBDD con información de usuario con propósito de autenticación y autorización.*
  - *NO Conector: permite a los bundles proveer nuevos y alternativos protocolos para javax.microedition.io.*
  - *Preferences Service: provee acceso a base de datos jerárquica de propiedades. Similar al registro de Windows.*
  - *Servicio HTTP: este servicio ejecuta un contenedor de servlets, los bundles pueden proveer servlets que son accesibles mediante HTTP.*
  - *Servicio UPnP: mapea dispositivos en una red UPnP al Registro de Servicios.*
  - *Wire Admin: permite la composición de diferentes servicios de acuerdo a una conexión.*

3.2.4 IMPLEMENTACIONES OSGi

--- --- --- --- --- --- --- ---

Las implementaciones de OSGi las podemos dividir en dos grupos para su clasificación:

-Implementaciones comerciales

-Implementaciones libres, en las que nos centraremos en Apache- Felix, Equinox y Knoplerfish, ya que serán las utilizadas en el presente proyecto.

Comerciales:

- *Makewave Knopflerfish Pro 2.0 (www.makewave.com) _comercial/libre*
- *ProSyst Software mBedded Server 6.0 (www.prosyst.com)*
- *Samsung OSGi R4 Solution (www.samsung.com)*

- *KT OSGi Service Platform (KOSP) 1.0 (http://www.kt.co.kr/)*
- *HitachiSoft SuperJ Engine Framework (http://hitachisoft.jp/)*

Libres:

- *JEFFREE: (Java Embedded Framework FREE) Aplicación de código abierto para las pasarelas residenciales que quedo abierta la licencia en el 2003 y es para la especificación OSGi 2.0.*
- *Oscar: (Open Service Container Arquitecture) Es unaherramienta para generar paquetes, se denomina Mangen, para funcionalidades con HTTP y JMX.*
- *Eclipse Equinox: framework OSGi usada en Eclipse (ver apartado de Tecnologías empleadas).* Conjunto de paquetes que implementan diversas funciones y servicios de OSGi para el funcionamiento de los sistemas basados en OSGi.
- ***Knopflerfish***

  En el caso de la integración de BitTorrent en OSGi, ha sido necesario instalar el plugin Knopflerfish. El objetivo es poder implementar las especificaciones OSGi de manera completa, y en este caso, se ha creado en lugar de un *project* un *bundle project* que como ya se explicará en el apartado de implementación de código, es lo que necesitamos ya que trabajamos con *bundles*. En definitiva, Knopflerfish es un desarrollo en código libre del framework especificado en OSGiR3. Facilita la experimentación de software orientado a componentes y servicios. **[21]**

  Presenta una interfaz intuitiva desde la que se nos permite instalar, desinstalar, iniciar, parar y actualizar componentes. A través del servicio Configuration Admin, es posible configurar los *bundles* que requieran una configuración cuando son desplegados. De esta manera, un *bundle* puede configurar a otro a través de este servicio.

- ***Apache Felix:***

  Apache es un servidor de páginas web. Surgió como un proyecto para desarrollar y mantener una fuente abierta HTTP para sistemas operativos como Unix,Windows. Es decir, que Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 (RFC 2616) y la noción de

sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. **[16]**

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. El objetivo de este proyecto es proporcionar un servidor seguro, eficiente y extensible que proporcione servicios HTTP en sintonía con los estándares HTTP actuales.

El servidor web Apache es donde empezó todo y, aunque ahora sólo es un proyecto entre muchos, se han seguido desarrollando nuevas y emocionantes características.Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. Apache tiene amplia aceptación en la red: en el 2005, Apache es el servidor HTTP más usado, siendo el servidor HTTP del 70% de los sitios web en el mundo y creciendo aún su cuota de mercado historia, unos 15 años.

Existen múltiples proyectos de este servidor. En el proyecto que nos vamos a centrar es en el Felix. Apache-Felix fue creado en para ofrecer una plataforma OSGi y se convirtió en una de las alternativas más importantes para la construcción de sistemas modulares en Java.

3.3 *PROTOCOLO BITTORRENT*

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

.Con BitTorrent podremos descargar el 30% del tráfico que se mueve por internet, como pueden ser películas, videos, software… En primer lugar se definirá qué es BitTorrent. Es un programa mezcla de P2P y FTP

A diferencia de los sistemas de intercambio de ficheros tradicionales, su principal objetivo es el proporcionar una forma eficiente de distribuir un mismo fichero a un gran grupo de personas, forzando a todos los que descargan un fichero a compartirlo también con otros. Primero se distribuye por medios convencionales un pequeño fichero con extensión .torrent. Este fichero es estático, por lo que a menudo se encuentra en páginas web o incluso se distribuye por correo electrónico. El fichero 'torrent' contiene la dirección de un "servidor de búsqueda", el cual se encarga de localizar posibles fuentes con el fichero o parte de él.

Este servidor realmente se encuentra centralizado y provee estadísticas acerca del número de transferencias, el número de nodos con una copia completa del fichero y el número de nodos que poseen sólo una porción del mismo.

El fichero o colección de ficheros deseado es descargado de las fuentes encontradas por el servidor de búsqueda y, al mismo tiempo que se realiza la descarga, se comienza a subir las partes disponibles del fichero a otras fuentes, utilizando el ancho de banda asignado a ello. Ya que la acción de compartir comienza incluso antes de completar la descarga de un fichero, cada nodo inevitablemente contribuye a la distribución de dicho fichero. El sistema se encarga de premiar a quienes compartan más, a mayor ancho de banda mayor el número de conexiones a nodos de descarga que se establecerán.

Cuando un usuario comienza la descarga de un fichero, BitTorrent no necesariamente comienza por el principio del fichero, sino que se baja por partes al azar. Luego los usuarios se conectan entre sí para bajar el fichero. Si entre los usuarios conectados se dispone de cada parte del fichero completo (aún estando desparramado), finalmente todos obtendrán una copia completa de él. Por supuesto, inicialmente alguien debe poseer el fichero completo para comenzar el proceso. Este método produce importantes mejoras en la velocidad de transferencia cuando muchos usuarios se conectan para bajar un mismo fichero.

Cuando no existan ya más nodos con el fichero completo ("semillas" o "seeds") conectados al servidor de búsqueda, existe la posibilidad de que el fichero no pueda ser completado.

Puesto que la base de BitTorrent son los sistemas P2P y FTP, se explicará cual es la función de dichos sistemas.

3.3.1 FTP(File Transfer Protocol)

Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en una arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

La ventaja que presenta este protocolo es la rápida velocidad de conexión.

En el 1969 un grupo de investigadores del MIT presentó la propuesta del primer "Protocolo para la transmisión de archivos en Internet", y su desarrollo se termina en 1985. Los niveles de red se estructuran según la figura 14.

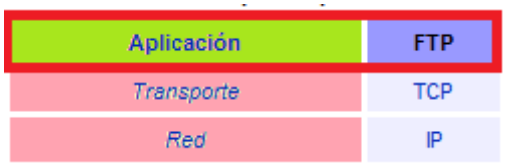

Figura 14: Niveles de Red

Modelo Cliente- Servidor FTP:

Un servidor FTP es un programa especial que se ejecuta en un equipo servidor normalmente conectado a Internet (aunque puede estar conectado a otros tipos de redes, LAN, MAN, etc.). Permite el intercambio de datos entre diferentes ordenadores.[27]

Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento web, en el que sus clientes utilizan el servicio para subir sus páginas web y sus archivos correspondientes; o como servidor de backup (copia de seguridad) de los archivos importantes que pueda tener una empresa.

Un cliente FTP es un programa que se instala en el ordenador del usuario, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos.

Para utilizar un cliente FTP, se necesita conocer el nombre del archivo, el ordenador en que reside (servidor, en el caso de descarga de archivos), el ordenador al que se quiere transferir el archivo (en caso de querer subirlo nosotros al servidor), y la carpeta en la que se encuentra.

Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos, incluyendo Windows, DOS, Linux y Unix. Sin embargo, hay disponibles clientes con opciones añadidas e interfaz gráfica.

3.3.2 P2P(Peer To Peer)

--- --- --- --- --- --- --- ---

El P2P es un modelo de comunicaciones en el que cada equipo tiene las mismas posibilidades para iniciar una comunicación con otro equipo. En la figura 15, vemos como todos los ordenadores están conectados entre sí. En Internet, los sistemas peer to peer son definidos como una red que permite el acceso a ficheros de otros ordenadores.

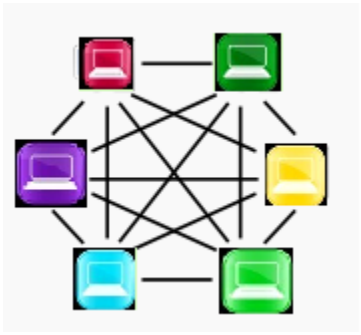

Figura 15: Modelo P2P

Estas redes se basan en la filosofía de que todos los usuarios deben compartir información, y el que más comparta más privilegios tendrá y más acceso disponible de manera más rápida al contenido compartido. Esta filosofía asegura la disponibilidad del contenido compartido.

Una vez definidas las redes P2P y el protocolo FTP, se procederá con la descripción del protocolo BitTorrent.

BitTorrent no incluye ningún mecanismo de búsqueda de archivos. Los usuarios de BitTorrent deberán localizar por sus propios medios los archivos torrent que necesita el protocolo. Normalmente, estos archivos pueden descargarse desde las páginas web que publican grandes archivos (como las distribuciones GNU/Linux) o desde índices web de búsqueda (como The Pirate Bay, Ktorrents o Bitgle).

3.3.3 ESTRUCTURA BITTORRENT

--- --- --- --- --- --- --- ---

Los equipos necesarios se componen en tres partes: Tracker, Seeds y Peers. Cada equipo o conjunto de equipos tiene una función determinada, y siguen el esquema de la figura 16[10]:

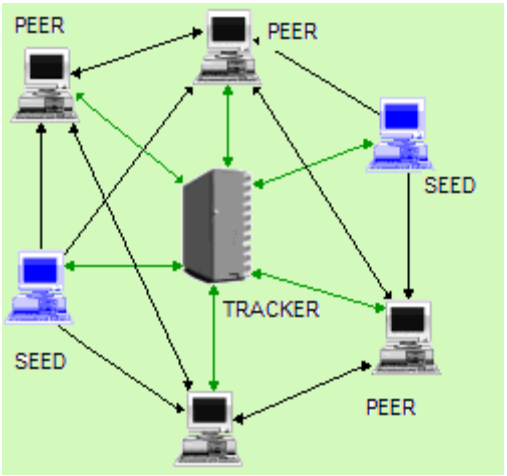

Figura 16: Estructura Bittorrent

- **TRACKER**: Es el único ordenador que conoce dónde se encuentran todos los usuarios y dónde todos los usuarios se conectaran para poder encontrar al resto de usuarios y conocer cuando se conectan nuevos usuarios. Podemos ver esta unión por las flechas de color verde por la que solo circula un poco de información para realizar la gestión de usuarios.

  En el caso de que el tracker se caiga, si alguien intenta comenzar una descarga, no podrá conectarse con el enjambre de usuarios. Su disponibilidad es imprescindible para poder comunicarse con el resto de usuarios.

  Además, los trackers pueden ser públicos o privados. Los privados requieren que los peers sean usuarios registrados de un sitio web, mientras que en los públicos cualquiera puede comunicarse con ellos. Los trackers privados generalmente guardan las estadísticas de tráfico de cada usuario y utilizan un sistema de porcentajes que permite saber si el usuario comparte o no los datos que haya descargado o esté descargando. Muchos de ellos suelen expulsar a los usuarios que tienen un porcentaje bajo, ya que al no compartir no colaboran con la red.

- **SEEDS**: Son los únicos ordenadores que tienen el 100% del archivo que queremos descargar. Estos usuarios envían el archivo al resto de usuarios. Como podemos observar en el esquema tan solo envían el archivo y no reciben nada.

- **PEERS:** Son los ordenadores que se dedican a descargar los archivos de todos los peers y, a la vez, suben el trozo que tienen de archivo a otros peers. Aunque en el esquema no se da el caso puede ser que un peer no esté conectado a ningún seed, incluso podría ocurrir que no hubiera seeds. Hay que tener en cuenta que cuanto más ancho de banda se comparte, más ancho de banda se recibe. O lo que es lo mismo, si queremos recibir el archivo rápidamente tendremos también que subir muchos datos (filosofía redes P2P).

3.3.4 TRANSFERENCIA Y CODIFICACIÓN DE ARCHIVOS .torrent

--- --- --- --- --- --- --- ---

Transferencia

El usuario que decide colgar un archivo, primero crea el archivo .torrent, el cual llevará toda la información para que pueda ser interpretado por el tracker. Este archivo será enviado al servidor principal, es decir, al tracker, que una vez que lo interpreta, indicará como lugar de descarga donde tenemos el original. El usuario que ha decidido colgarlo, se convertirá en un seed, ya que tendrá el total del archivo descargado puesto que lo ha subido él.

Un archivo torrent es dividido por el tracker en trozos dependiendo del total del archivo, de manera que hasta que no son completadas todas las divisiones no tendremos el 100% del archivo. Un .torrent solo podrá ser descartado por el tracker.

Debido a la filosofía P2P comentada anteriormente, es importante que los usuarios sigan compartiendo a pesar de que tengan el total del archivo, ya que se necesita mantener el funcionamiento de la red. Cuantos más Peers y Seeds, más rápidas serán las descargas.

Si el número máximo de Peers al que vamos a subir datos simultáneamente son cuatro, es decir, tener configurado cuatro conexiones de subida, las 3 conexiones primeras son usadas para enviar al peer que más datos haya enviado en los últimos 10 segundos y esté interesado en las pieces que disponemos, como se puede ver en los 3 círculos en verde de la figura 17.

NOTA: Los pieces son los trozos o bloques en los que se divide un archivo. Estos pieces no tendrán todas el mismo tamaño.

Optimistic Unchoke	IP	Up	Interested	Choking	Down	Downloaded	Uploaded	Completed
	1.1.1.1			*		5.34 MiB	6.12 MiB	99.2%
	1.1.1.2		*	*		2.12 MiB	2.01 MiB	20.3%
	1.1.1.3	4 kb/s	*		5 kb/s	30.37 MiB	26.52 MiB	70.8%
	1.1.1.4		*	*		0.31 MiB	0.50 MiB	4.2%
	1.1.1.5			*	6 kb/s	4.74 MiB	2.90 MiB	100.0%
	1.1.1.6	3 kb/s	*		4 kb/s	12.12 MiB	7.84 MiB	85.4%
	1.1.1.7	3 kb/s	*		2 kb/s	20.50 MiB	17.12 MiB	69.4%
	1.1.1.8			*		1.21 MiB	1.84 MiB	100.0%
*	1.1.1.9	2 kb/s	*			1.31 MiB	2.56 MiB	74.7%
	1.1.1.10		*	*	1 kb/s	3.32 MiB	4.35 MiB	9.7%

Figura 17: Descarga del algoritmo unchoke

El usuario con ip 1.1.1.5 nos manda datos a una velocidad mayor que el resto usuarios, pero no se le envía nada ya que como se puede ver en los círculos verdes de la figura 18, contiene el 100% del archivo y no está interesado en las pieces de las que disponemos porque se trata de un Seed.

La conexión restante es utilizada para la técnica Optimistic Unchoke(ver sección de algoritmos), que consiste en enviar datos durante 30 segundos a un usuario, luego a otro y así de manera que todos sean beneficiados. Estos usuarios, son marcados para señalar que han tenido el privilegio de la técnica. Con ello, se consigue que todos los usuarios tengan posibilidad de obtener un piece o trozo de archivo y para comprobar si hay otro usuario con el que intercambiar a mayor velocidad.

Optimistic Unchoke	IP	Up	Interested	Choking	Down	Downloaded	Uploaded	Completed
	1.1.1.1			*		5.34 MiB	6.12 MiB	99.2%
	1.1.1.2		*	*		2.12 MiB	2.01 MiB	20.3%
	1.1.1.3	4 kb/s	*		5 kb/s	30.37 MiB	26.52 MiB	70.8%
	1.1.1.4		*	*		0.31 MiB	0.50 MiB	4.2%
	1.1.1.5			*	6 kb/s	4.74 MiB	2.90 MiB	100.0%
	1.1.1.6	3 kb/s	*		4 kb/s	12.12 MiB	7.84 MiB	85.4%
	1.1.1.7	3 kb/s	*		2 kb/s	20.50 MiB	17.12 MiB	69.4%
	1.1.1.8			*		1.21 MiB	1.84 MiB	100.0%
*	1.1.1.9	2 kb/s	*			1.31MiB	2.56 MiB	74.7%
	1.1.1.10		*	*	1 kb/s	3.32 MiB	4.35 MiB	9.7%

Figura 18: Descarga del algoritmo unchoke

En cuanto a la velocidad de subida y de descarga, no son exactamente proporcionales, pero sí que cuanta mayor velocidad de subida, más rápida será la velocidad de descarga al poder conectar con los usuarios mejores. Además, hay que tener en cuenta que la velocidad de descarga depende de los usuarios conectados a la red BitTorrent, por ello, no descargamos a la velocidad de la línea. Ambas velocidades, subida y descarga, deben ser aproximadamente iguales. Este protocolo no se diseñó para realizar varias descargas a la vez, por lo que cuantas menos descargas simultaneas mejor.

Codificación interna

Los archivos .torrent contienen información acerca del archivo que queremos bajar. Esta información está codificada mediante Bencoding.

Al abrir con un editor de texto un archivo .torrent nos encontramos con un diccionario que contiene las siguientes claves:

- *info*: Un diccionario que describe los archivos del torrent. Puede tener una u otra estructura dependiendo de si el torrent es para bajar un archivo o varios archivos con una jerarquía de directorios.
- *announce*: cadena que representa la URL del *tracker*
- announce-list: (lista de cadenas opcional). Se usa para representar listas de trackers alternativos. Es una extensión a la especificación original.

- *creation date*: (entero opcional) La fecha de creación del torrent en formato de época UNIX.
- *comment*: (cadena opcional) Campo libre para el creador del torrent.

- *created by*: (cadena opcional) Nombre y versión del programa usado para crear el archivo torrent.

El diccionario info que acabamos de citar contiene a su vez las siguientes claves:

- *name*: (cadena) El nombre del archivo o directorio donde se almacenarán los archivos.
- piece length: Como dijimos en la introducción, el archivo que queremos compartir es dividido en piezas. Este parámetro es un entero que representa el número de bytes de cada pieza. Piezas demasiado grandes causan ineficiencia y piezas demasiado pequeñas forman un archivo .torrent más pesado. Actualmente se aconseja fijar el tamaño de cada pieza en 512 KB o menos para archivos de varios GBs.
- *pieces*: Cadena que representa la concatenación de la lista de claves hash de cada parte del fichero compartido. Las claves hash son generadas utilizando SHA-1 con un resumen de 160 bits y un tamaño máximo por parte de 2^64 bits. Este conjunto de claves se utiliza como mecanismo para asegurar la integridad y consistencia de una parte, una vez ha sido completada la descarga de dicha parte.
- *private*: (opcional). Es un entero que puede tener valores 0 ó 1 y que indica si se pueden buscar *peers* fuera de los *trackers* explícitamente descritos en la metainformación o no.
- *length*: (entero) Longitud del archivo en bytes.
- *files*: Sólo aparecerá en el caso de que sea un torrent multi archivo. Es una lista de diccionarios (uno para cada archivo, pero con una estructura diferente a info). Cada uno de estos diccionarios contendrá a su vez información sobre la longitud del archivo, la suma MD5 y una ruta (path) en donde debe ubicarse el archivo en la jerarquía de directorios.

3.3.5 ALGORITMOS PARA LA COMPARTICIÓN

En esta sección se van a explicar detalladamente las reglas según las cuales se elige a uno u otro usuario para compartir partes del archivo y qué partes son las que se transmiten.

En primer lugar vamos a describir unos cuántos términos importantes:

- *Piezas y bloques*. Los archivos transmitidos usando Bittorrent se dividen en piezas y éstas a su vez se dividen en bloques. Los bloques son la unidad de transmisión en la red, pero las piezas parcialmente recibidas no pueden ser servidas por un peer hasta que estén completas, es decir que se tengan todos sus bloques.
- *Interested*. Se dice que un peer A está interesado en el peer B (A está en el estado Interested) cuando el peer B tiene piezas que el peer A no tiene. Por el contrario, el peer A no está interesado en el peer B cuando el peer B solo tiene un subconjunto de las piezas del peer A.
- *Choked*. Se dice que el peer A bloquea al peer B (B está en el estado Choked) cuando el peer A decide no enviar piezas al peer B. Por el contrario, se dice que el peer A desbloquea al peer B cuando el peer A decide enviar piezas al peer B.
- *Conjunto de peers*. Cada peer mantiene una lista de los *peers* que conoce.
- *Peers locales y remotos*. Se llama peer local al par que está ejecutando el cliente Bittorrent y *peers* remotos a los pares que están en el conjunto de *peers* del peer local.
- *Conjunto de peers activos.* El peer A sólo puede enviar datos a un subconjunto de su conjunto de *peers*. A este conjunto se le llama conjunto de *peers* activos. El algoritmo de bloqueo , que veremos más adelante, determina los *peers* que formarán parte del conjunto de *peers* activos. Solamente los *peers* que están desbloqueados por el peer local e interesados en él son parte del conjunto de *peers* activos.
- *Piezas más raras y conjunto de las piezas más raras*. Las piezas más raras son las que tienen menos número de copias en el conjunto de *peers*. En el caso de que la pieza menos replicada en el conjunto de *peers* tenga m copias, entonces todas las piezas con m copias forman el conjunto de las piezas más raras.

Rarest First Algorithm

Este algoritmo define la estrategia usada por el protocolo Bittorrent para seleccionar la siguiente pieza a descargar. Cada par mantiene una lista del número de copias de cada pieza en su conjunto de pares y usa esta información para definir su conjunto de las piezas más

raras. Sea m el número de copias de la pieza más rara, entonces la posición de cada pieza con m copias en el conjunto de peers es añadida al conjunto de las piezas más raras. Cada peer selecciona aleatoriamente la siguiente pieza para bajar de su conjunto de las piezas más raras.

El comportamiento de este algoritmo puede ser modificado por 3 causas:

1. Si un peer ha bajado menos de 4 piezas, éste elige aleatoriamente la próxima pieza para bajar. Una vez que se han descargado estas 4 piezas, el algoritmo funciona de la manera descrita anteriormente. La razón de este comportamiento inicial es permitir que un peer baje sus primeras piezas muy rápidamente, ya que es importante tener algunas piezas para empezar a intercambiar en el Algoritmo de Bloqueo. De hecho, una pieza elegida aleatoriamente tiene muchas más copias que las piezas más raras, así que probablemente el tiempo de bajada será menor seleccionándola aleatoriamente.

2. La segunda causa es que cuando uno de los bloques de una pieza se ha pedido, los otros bloques de la misma pieza se piden con la prioridad más alta. La razón de este comportamiento es completar la descarga de una pieza completa lo antes posible, dado que sólo las piezas completas pueden ser enviadas.

3. La última causa es el modo de fin de juego (End Game Mode). Este modo empieza muy al final de la descarga, cuando el peer pide todos los bloques que todavía no han sido recibidos a todos los *peers* de su conjunto de *peers* que tienen esos bloques. Cada vez que se recibe un bloque, el peer cancela la petición para el bloque recibido a todos los *peers* en su conjunto de *peers* que tienen la petición activa.

Choke Algorithm

Este algoritmo define la estrategia usada por el protocolo Bittorrent para seleccionar el siguiente peer con el que interactuar. Se usa para garantizar un buen ratio subida/bajada entre los peers. Por ejemplo los “free riders”, pares que nunca suben, deben ser penalizados. El algoritmo se describe desde el punto de vista del peer local, así que “interesado” significa interesado en el peer local y “bloqueado” significa bloqueado por el peer local. El algoritmo funciona así:

1. Como máximo 4 *peers* remotos pueden estar desbloqueados e interesados a la vez.

2. Cada 10 segundos, los *peers* remotos interesados se ordenan de acuerdo a su velocidad de bajada hacia el peer local y los 3 más rápidos son desbloqueados.

3. Cada 30 segundos, un peer interesado adicional se desbloquea aleatoriamente. Esto se llama “Desbloqueo Optimista” (Optimistic Unchoke).

El Desbloqueo Optimista tiene dos objetivos. Permite evaluar la capacidad de bajada de nuevos peers en el conjunto de peers y también posibilita que los peers que no tienen ninguna pieza que compartir puedan obtener su primera pieza.

3.4 *HERRAMIENTAS Y TECNOLOGÍAS USADAS*

Para el desarrollo de este proyecto es fundamental tener profundos conocimientos del lenguaje de programación JAVA.

Java es una plataforma virtual de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales ("Diferentes plataformas"). [15]

La plataforma Java consta de las siguientes partes:

1. El lenguaje de programación, mismo.

2. La máquina virtual de Java o JRE, que permite la portabilidad en ejecución.

3. El API Java, una biblioteca estándar para el lenguaje.

Java permite escribir programas de interfaz gráfica o textual. También se pueden correr programas de manera incorporada o incrustada en los navegadores web de Internet en forma de Java applets, aunque no llegó a popularizarse como se esperaba en un principio.

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado bytecode, que luego son interpretados por una máquina virtual (JVM). Esta última sirve como una plataforma de abstracción entre la máquina y el lenguaje permitiendo que se pueda "escribir el programa una vez, y correrlo en cualquier lado". También existen compiladores nativos de Java, tanto software libre como no libre. El compilador GCC de GNU compila Java a código de máquina con algunas limitaciones al año 2002. [16]

Con la evolución de las diferentes versiones, no sólo se han producido cambios en el lenguaje, sino que se han producido cambios mucho más importantes en sus bibliotecas asociadas, que han pasado de unos pocos cientos en Java 1.0, a más de tres mil en Java 5.0. En particular, se han añadido APIs completamente nuevas, tales como Swing y Java2D.

La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Pero no se queda ahí, Java se desarrolla dispositivos además del ordenador como móviles, agendas y en general para cualquier dispositivo. [26]

Para la implementación de código, el entorno de desarrollo utilizado es eclipse, entorno de desarrollo integrado de código abierto multiplataforma. Eclipse es una plataforma de desarrollo basada en Java que desarrolló IBM. En sí mismo es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados conocidos como plug-ings. Estos plug-ings, pueden ser para desarrollo Java, C, C++.

Dependiendo del tipo de tarea proporciona una serie de funcionalidades. Permite organizar los ficheros en forma de proyecto, y cuando por ejemplo se crea un proyecto Java, eclipse abre automáticamente toda la perspectiva Java, creándose en el directorio especificado el .project y .classpath

Eclipse puede compatibilizarse con OSGI importando el jar correspondiente en la pestaña bulit path [22]. En la figura 19, se ve como es la interfaz de este editor.

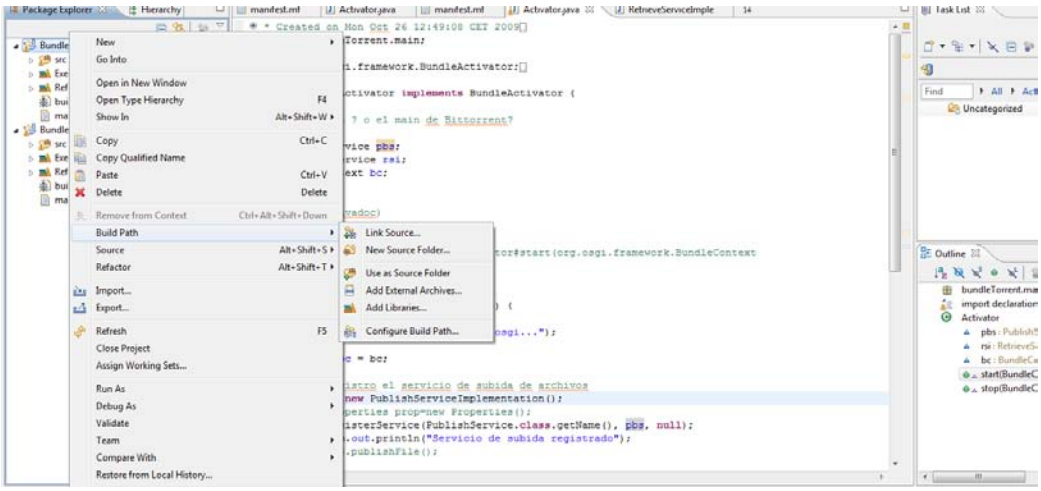

Figura 19: Interfaz eclipse

Los programas pueden ser ejecutados dentro de eclipse. Dentro del entorno de desarrollo de Eclipse se pueden depurar programas desarrollados en Java. [23]

Como herramientas interesantes, el editor de Java ofrece correcciones a problemas encontrados mientras se escribe el código y tras compilar. El editor muestra que existen propuestas para la corrección de un problema o aviso mediante una bombilla visible en la

parte izquierda del editor. Si se pulsa con el botón izquierdo sobre la bombilla se muestran las propuestas para el problema en la posición del cursor.

Desde un proyecto se pueden exportar todos o algunos de los ficheros que lo conforman. Para ello en la vista Navigator se pulsa sobre el botón derecho y aparece un menú, se selecciona Export y aparece una ventana en la que podemos indicar cómo se va a exportar (un fichero zip, tal cual aparecen en el proyecto, ...). La siguiente ventana sirve para seleccionar los ficheros que se desean exportar y a dónde.

Si colocamos el ratón sobre un método (sin pulsar) se nos muestra la declaración del método (qué devuelve y qué parámetros acepta), como se ve en la figura 20. Si colocamos el ratón (sin pulsar) sobre una variable aparece información sobre el tipo de la variable. Al escribir código podemos pulsar Ctrl + espacio y nos aparece un menú con posibles formas de finalizar la sentencia que se está escribiendo.

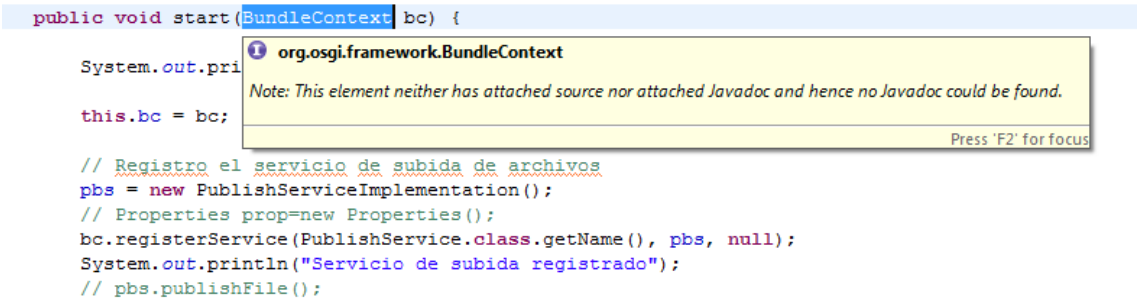

Figura 20: Información del método

4. ESTUDIO DE VIABILIDAD

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

El estudio de viabilidad busca analizar la viabilidad económica, organizativa y técnica del proyecto. Para ello, se realiza un estudio de las necesidades y se estudian los casos de utilización. En definitiva, todo plan de negocio, tiene como objetivo tomar una decisión.

4.1 LA IDEA

La idea de negocio es realizar una aplicación que permita compartir archivos con otros usuarios a través de una plataforma domótica y basándose en el protocolo BitTorrent. Para desarrollar esta idea, la dividiremos en cuatro partes fundamentales:

Se dispondrá de un servidor compatible con el protocolo BitTorrent, de manera que cuando los usuarios se conecten a él puedan acceder a los archivos e información compartida. Utilizará el protocolo HTTP.

Por otro lado, dispondremos de la aplicación Torrent en sí, que contiene la manera de transferir los archivos, es decir el protocolo propiamente dicho.

Se creará la aplicación Client, que podrá cargar archivos para su compartición de manera sencilla y dinámica, así como realizar la descarga de los ficheros.

Para este último punto, que es el que afecta directamente al cliente, se creará una sencilla implementación de comandos para el dinamismo de la compartición.

Todos los clientes conectados al servidor podrán transferir o recibir ficheros de manera sencilla, desde diferentes lugares.

4.2 ANÁLISIS DEL ENTORNO

Dado el entorno social y económico actual, será idóneo el ahorro en costes. Además la disminución de ventas y el incremento de la competencia como consecuencia, hacen que el producto tenga que ser de lo más atractivo para nuestros clientes. En la domótica, tenemos la ventaja de que es un entorno novedoso y atractivo.

Gracias a la estandarización de la tecnología que se está llevando a cabo, la domótica evoluciona favorablemente, por lo que esta aplicación o propiamente el protocolo BitTorrent será una gran ventaja al poderla integrarla en un dispositivo domótico.

Con la ventaja que presenta la modularidad que presenta OSGi, como la actualización de versiones y la necesidad de un único dispositivo (pasarela) al que pueden conectarse los

diferentes dispositivos domóticos, se permite el ahorro de costes necesario, en los tiempos que corren, frente a la desventaja debido a la crisis económica que atraviesa el país.

4.3 ESTUDIO DE LA COMPETENCIA

Para la transferencia de archivos, se fueron creando diferentes programas hasta día de hoy, y muchos de ellos han tenido una gran aceptación, ya que para el usuario es muy cómodo poder disponer en su ordenador de ciertos ficheros sin necesidad de ir a comprarlos.

A nivel de legalidad, España es uno de los países con mayor número de usuarios en redes P2P, y en algunos sectores de la sociedad, la descarga de ficheros protegidos es considerado un acto de piratería, mientras que otros defienden la legalidad de esta acción basándose en el derecho de los usuarios a realizar copias privadas. Este debate está servido a día de hoy y desde hace unos años atrás.

A nivel de competencia, en el mercado, los programas que más fuerza cobran son los siguientes:

-eMule: soporta la red eDonkey y no permite encriptación.
-Ares: buenas velocidades, permite autoencriptado en los paquetes de datos entre los clientes.
-Azureus, soporte de red BitTorrent.
-BitCommet, soporte de red BitTorrent.
-µTorrent, soporte de red BitTorrent.
-Kazaa, soporte de red Kazaa.
-eXeem, que utiliza al igual que Azureus o BitCommet el protocolo BitTorrent.

La diferencia fundamental entre todos ellos, son las redes que soportan y los protocolos que emplean. Muchos de los primeros en las listas de uso, emplean BitTorrent. Aun así, hay una gran competencia, y la integración propuesta en el presente proyecto, tendrá un gran tirón en el entorno domótico, puesto que el protocolo BitTorrent está en la lista de los diez primeros más usados.

4.4 CONCLUSIONES

Como conclusión de la viabilidad del proyecto, se observa que contamos con la ventaja de que en la competencia no disponen de la integración realizada, por lo que es algo novedoso.

Además, se permiten posibles modificaciones de gestión para el cliente, pudiendo incorporar una interfaz visual más intuitiva y atractiva para el cliente. Para ello, bastará con tener instalado el bundle del protocolo torrent e incluir un nuevo bundleClient o modificar el existente. Diferentes aplicaciones pueden hacer uso de estos servicios.

La desventaja que se nos presenta, es el alto despliegue inicial, puesto que tendrá que darse a conocer argumentando las ventajas que presenta OSGi, ya que la mayoría de usuarios de estos programas P2P están acostumbrados a utilizarlos sin necesidad de tener la pasarela en su casa, por lo que se considera que los usuarios de este productos serán muy selectos, ya que solo interesará para clientes que ya dispongan de más dispositivos domóticos en su hogar. Por otro lado, el coste de pruebas de Sofware se llevará parte de la inversión inicial.

Sacadas estas conclusiones, podremos vender nuestro producto como algo único que no posee competencia y algo innovador dentro de una pasarela residencial.

4.5 PRESUPUESTO

Esta sección, cuantifica el esfuerzo realizado dándole una proyección económica.

El salario medio será de 15€/hora en relación a un Ingeniero Técnico de Telecomunicaciones recién titulado. Más adelante, se mostrará una tabla con las tareas realizadas, el tiempo empleado en cada una de ellas y el presupuesto que suponen en función de la dedicación y teniendo en cuenta una jornada laboral de 7h/día.

La definición de las tareas, se marca a través de las siguientes fases, mostrándose en la tabla 1:

- *Fase de estudio*: se estudia el modelo de negocio observando los requisitos necesarios. Se hade una planificación del proyecto y se elabora un presupuesto.
- *Fase de análisis y diseño*: en esta etapa se incluye el análisis y diseño, es la etapa de ingeniería “dura”.
- *Fase de implementación*: se construye el sistema en sí, probándolo en profundidad para la posterior fase de pruebas. También se desarrolla un manual de usuario en el caso de que sea necesario

- *Fase de pruebas o testing*: Se realizan las últimas pruebas antes de la aceptación del proyecto.

FASE	TAREA	DURACIÓN (Días)	COSTE(€)
1.Fase de Estudio		**23**	**2415**
	1.1 Investigación	20	2100
	1.2 Viabilidad	3	315
2.Fase de Análisis y Diseño		**16**	**1680**
	2.1 Análisis y Diseño	16	1680
	2.1.1 Análisis del protocolo BitTorrent	4	420
	2.1.2 Adaptación del código inicial del protocolo a nuestras necesidades	6	630
	2.1.3 Diseño de integración:	6	630
	2.1.3.1 Tracker	2	210
	2.1.3.1 BitTorrent	2	210
	2.1.3.1 Uso por parte de los clientes- Subida y Descarga	2	210
3. Fase de Implementación		**40**	**4200**
	3.1 Implementación	40	4200
	3.1.1 Tracker	2	210
	3.1.2 BitTorrent	15	1575
	3.1.3 Uso por parte de los clientes- Subida y Descarga	23	2415
4. Fase de Pruebas		**3**	**315**
5. Documentación		**95**	**(*)560**
TOTAL (*)		**95**	**9.170 €**

Tabla 1: Presupuesto sin costes

(*) La documentación se ha ido realizando según ha ido avanzando el proyecto, por lo que el coste se ha calculado descontando los días empleados en las diferentes etapas y contando únicamente los días posteriores en los que se concluido todo el contenido de la memoria.

(*) Al total falta añadirle los costes directos e indirectos.

Costes Directos e Indirectos.

A continuación, se añaden los costes directos e indirectos que suponen llevar a cabo el proyecto. Es decir, se desarrolla una aproximación de los costes de los recursos necesarios para completar las actividades del proyecto.

Costes directos: Se entiende por coste directo aquel que puede identificarse con productos, costes que corresponden al material y equipo necesarios comprometidos directamente con la ejecución. En nuestro caso, estos costes vendrán determinados por el PC utilizado en la realización, que supone un coste de 1000€, y un coste adicional de 150€ por la implementación del proyecto en la pasarela OSGi.

Costes indirectos: Son los costes relativamente independientes del proyecto, como los producidos por las licencias de herramientas genéricas de software. Gracias a la utilización de programas de código libre, como java orientada a objetos, Apache-Felix y el código fuente disponible de BitTorrent, supondrá un ahorro notable. Además, tenemos que añadir costes ineludibles como la luz y conexión de banda ancha, que supondrán un coste del 5% del coste total del proyecto.

Amortización: La amortización es un término económico y contable, referido al proceso de distribución en el tiempo de un valor duradero. Teniendo en cuenta este término, el Hardware empleado, es decir, el PC, lo hemos considerado amortizable en 2 años, y en el caso de Software, se ha considerado que el gasto del sistema operativo viene intrínseco en la amortización del PC.

Cálculo total del presupuesto		TOTAL
Costes Directos	1PC*1000€*5meses/(2años*12meses)	208,33 €
Costes Indirectos	5%*9170€	458,5 €
Presupuesto (sin costes incluidos)	9.170 €	9.170 €
TOTAL		**9.836,83 €**

Tabla 2: Presupuesto con costes directos e indirectos

5. ANÁLISIS Y DISEÑO DEL SISTEMA

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

5.1 DEFINICIÓN DEL ENTORNO

En este apartado, se presenta el marco dónde se va a encontrar nuestra aplicación, de manera que se comprenda mejor este capítulo de análisis y diseño.

Todo el desarrollo de la aplicación estará programado en lenguaje JAVA. El protocolo BitTorrent, será integrado en la pasarela residencial OSGi mediante aplicaciones llamadas *bundles*. En la figura 21, se muestra como queda el entorno.

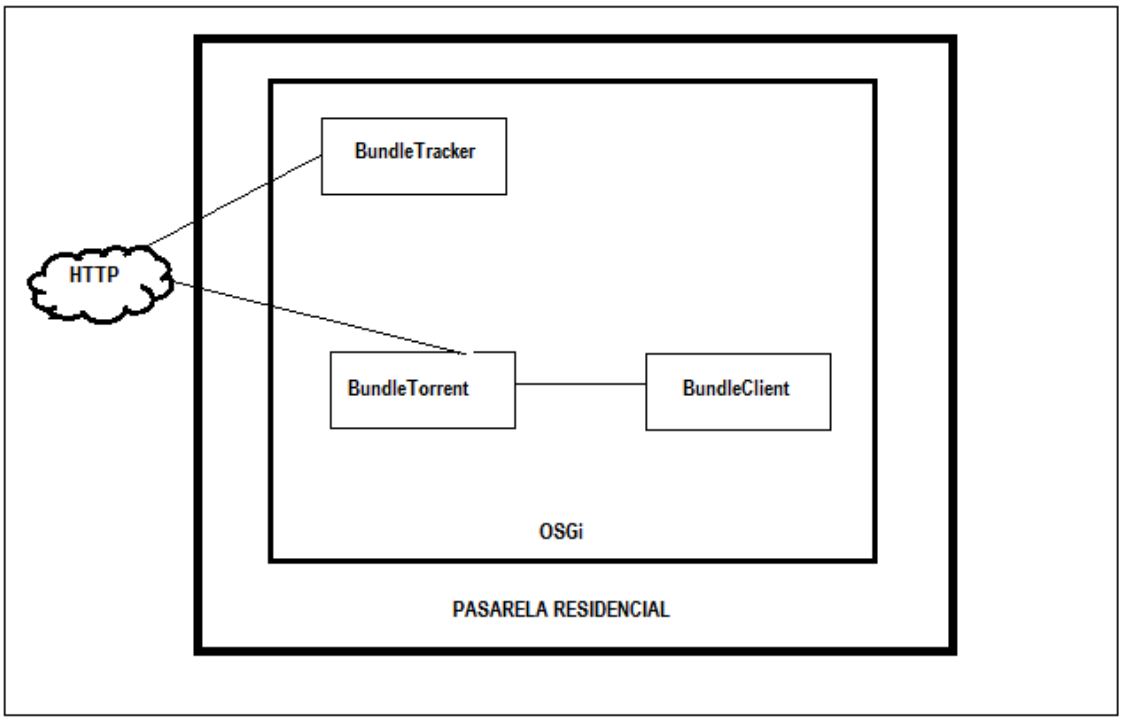

Figura 21: Definición del entorno

El objetivo de la pasarela residencial, es centralizar todos los dispositivos y controlar los servicios registrados en OSGi.

Como se ve en la figura 20, en OSGi se encontrarán instalados los siguientes bundles:

El bundle que implementa el servidor, BundleTracker: se conecta a través del protocolo HTTP y permite la transferencia de archivos. Además, posee de un archivo de configuración para guardar en la base de datos los torrents disponibles y realizar una actualización en la lista de peers.

El bundle que implementa el protocolo BitTorrent, en esta aplicación se registran los servicios de carga y descarga de archivos para el posterior uso del BundleClient.

El BundleClient, bundle del cliente, usa los servicios proporcionados por el BundleTorrent, y permite la interacción del usuario final con el programa completo.

Y una vez definido el entorno, destacar que el proyecto se encarga de permitir a un usuario final de la transferencia de archivos según el protocolo BitTorrent, a través del servidor tracker definido.

5.2 CASOS DE USO

Los casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios.

Por tanto, la especificación de los casos de uso del sistema, nos proporciona una descripción con los escenarios en los cuales los usuarios finales interaccionan con el sistema.

Los casos de uso que se pueden dar en nuestra aplicación se detallan en unas tablas auto explicativas que veremos a continuación. Primero se presenta un diagrama del conjunto de casos de uso.

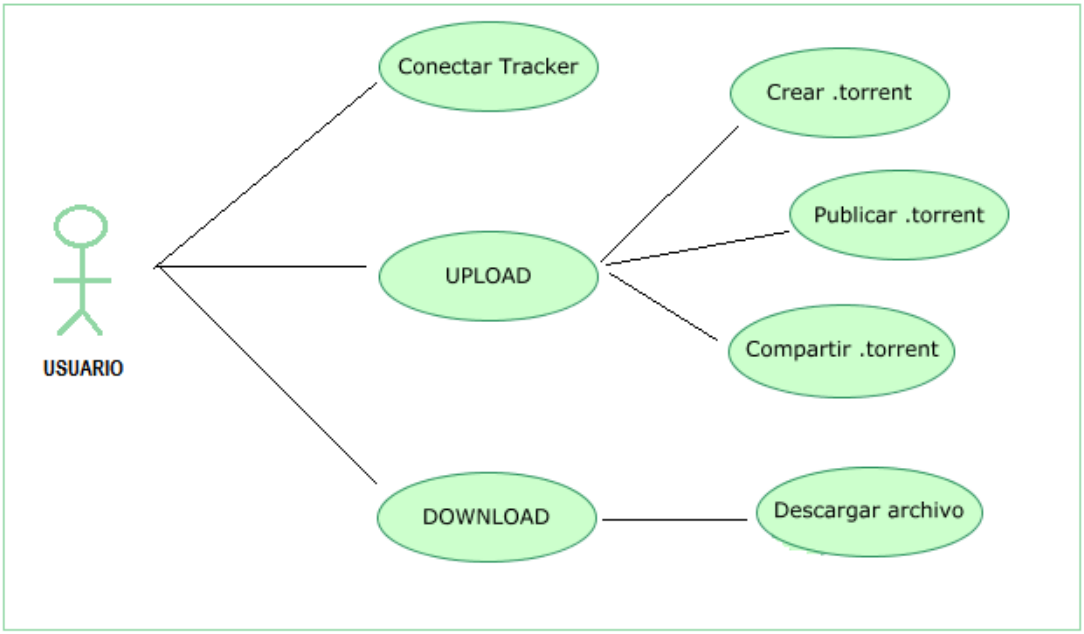

Figura 22: Diagrama de Casos de Uso del Usuario

Visto, el diagrama se presenta una tabla auto explicativa a modo de ejemplo:

IDENTIFICADOR	**CASO DE USO DE EJEMPLO**
Versión	1.0 (Mayo 2010)
Descripción	Descripción del requisito
Actores	Rol que adquiere el usuario que interacciona con el sistema
Objetivo	Servicio que el actor busca conseguir
Pre-Condiciones	Descripción del conjunto de estados del sistema anteriores a la ejecución del caso de uso
Post-Condiciones	Descripción del conjunto de estados del sistema posteriores a la ejecución del caso de uso
Escenario	Secuencia de acciones principales (información intercambiada) en la interacción del escenario básico

Tabla 3: Caso de uso de ejemplo

A continuación, se describe cada caso de uso atendiendo a la información de la tabla 3.

Cada caso de uso tendrá asociado un identificador único que los diferencia del resto de de casos, con la siguiente estructura:

CU: serán las dos primeras letras comunes para todos los casos ya que indica **C**aso de **U**so.

U: indica **U**suario final.

Numero de caso de uso: número dentro de la clasificación.

*Ejemplo***: CU-U00**: Caso de uso del usuario número 0.

CU-U01	**Conexión del Tracker**
Versión	1.0 (Mayo 2010)
Descripción	Se conectará el servidor a través del comando conectaTB
Actores	Usuario que desee conectar un servidor
Objetivo	Que se pueda realizar la transferencia de archivos
Pre-Condiciones	El servidor no estará escuchando
Post-Condiciones	El servidor estará listo para transferir archivos, añadir los *torrents* a la base de datos y actualizar la lista de peers
Escenario	El usuario introduce el comando conectaBT, se lee un fichero de configuración necesario para su ejecución y el *tracker* se conecta por http a la dirección indicada en los parámetros de configuración.

Tabla 4: CU-U01: Conexión del Tracker

CU-U02	**UPLOAD**
Versión	1.0 (Mayo 2010)
Descripción	Engloba las interacciones para que un usuario pueda subir un archivo
Actores	Usuario que desee compartir un archivo
Objetivo	Que se pueda cargar un archivo en un servidor para estar disponible para otros usuarios
Pre-Condiciones	Será necesaria la conexión del *tracker* y la instalación y ejecución del *Torrent.jar*
Post-Condiciones	El servicio *TorrentPublishService* queda registrado para poder ser usado por el cliente.
Escenario	Instalación del bundleTorrent. Existen tres escenarios de uso de este servicio correspondientes a las tablas **CU-U12,CU-U22** y **CU-U32**

Tabla 5: CU-U02: UPLOAD

CU-U12	**Crear .torrent**
Versión	1.0 (Mayo 2010)
Descripción	Creación del archivo *.torrent* a través del comando **CT**
Actores	Usuario que desee compartir un archivo
Objetivo	Crear un archivo *.torrent* que dispone de la información del fichero original
Pre-Condiciones	Será necesaria la conexión del *tracker* y la instalación y ejecución del *Torrent.jar*
Post-Condiciones	El *.torrent* queda anunciado en el *tracker*
Escenario	El usuario introduce el comando: **CT <torrentName> <fileName> <author> <comment>**. El *torrent* del archivo *fileName* se crea con el nombre *torrentName*. Este archivo *.torrent* lleva un código hash correspondiente al archivo, el comentario, el autor, el número de trozos en los que va a dividirse el archivo y la URL del *tracker* donde es anunciado. El *.torrent* queda anunciado en el servidor.

Tabla 6: CU-U12: Crear .torrent

CU-U22	**Publicar .torrent**
Versión	1.0 (Mayo 2010)
Descripción	Publicación del archivo *.torrent* en el servidor con el comando **PT**
Actores	Usuario que desee compartir un archivo
Objetivo	Publicar un archivo *.torrent* que dispone de la información del fichero original
Pre-Condiciones	Será necesaria la conexión del *tracker*, instalación/ejecución del *Torrent.jar* y la creación del archivo *.torrent*
Post-Condiciones	El *.torrent* queda publicado en el *tracker*
Escenario	El usuario introduce el siguiente comando: **PT <torrentName> <comment>**. El *.torrent* queda publicado en el servidor. Existe la posibilidad de introducir un usuario y contraseña para que no esté accesible a cualquier usuario, pero actualmente esta opción esta desactivada.

Tabla 7: CU-U22: Publicar .torrent

CU-U32	**Compartir .torrent**
Versión	1.0 (Mayo 2010)
Descripción	A través del comando **PT** el *.torrent* queda subido en el servidor y disponible para ser descargado
Actores	Usuario que desee compartir un archivo
Objetivo	Compartir un archivo *.torrent* que dispone de la información del fichero original
Pre-Condiciones	Será necesaria la conexión del *tracker*, instalación/ejecución del *Torrent.jar* y la creación del archivo *.torrent*
Post-Condiciones	El *.torrent* queda compartido
Escenario	El usuario introduce el siguiente comando: **PT <torrentName> <comment>** El *.torrent* queda disponible en el servidor para el paso posterior de descarga.

Tabla 8: CU-U32: Compartir .torrent

CU-U03	**DOWNLOAD**
Versión	1.0 (Mayo 2010)
Descripción	Engloba las interacciones para que un usuario pueda descargar un archivo
Actores	Usuario que desee descargarse un archivo
Objetivo	Descarga de un archivo disponible en el *tracker*
Pre-Condiciones	Será necesaria la conexión del *tracker*, instalación/ejecución del *Torrent.jar*
Post-Condiciones	El servicio *TorrentDownloadService* queda registrado para poder ser usado por el cliente.
Escenario	Instalación del bundleTorrent. Existe un escenario de uso del servicio correspondiente a la tabla **CU-U13**

Tabla 9: CU-U03: DOWNLOAD

CU-U13	**Descargar archivo**
Versión	1.0 (Mayo 2010)
Descripción	Descarga de un archivo disponible en el *tracker* a través del comando **DT**
Actores	Usuario que desee descargarse un archivo
Objetivo	Descarga de un archivo disponible en el *tracker*
Pre-Condiciones	Será necesaria la conexión del *tracker*, instalación/ejecución del Torrent.jar y creación/publicación-compartición del *.torrent*
Post-Condiciones	El archivo queda descargado
Escenario	El usuario introduce el comando **DT <torrentName> <downloadFolder>.** El archivo es descargado en la carpeta que indica el usuario del directorio C:/TORRENT/DOWNLOAD/

Tabla 10: CU-U13: Descargar archivo

5.1 ESPECIFICACIONES DE LOS REQUISITOS DE SOFWARE

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

En la especificación de requisitos de software (ERS), se realiza una descripción completa del comportamiento de servicio BitTorrent en la plataforma OSGi.

Para el desarrollo de esta fase, tendremos en cuenta requisitos funcionales, que responden a los casos de uso, es decir, al comportamiento interno del software y requisitos no funcionales, que son lo que imponen restricciones en el diseño o implementación.

Al igual que en los casos de uso, cada requisito tendrá asociado un identificador único que lo diferencia del resto. El formato será el siguiente:

Tipo de Requisito: **RF, RNF** Requisito Funcional, Requisito No Funcional

Número de requisito: indica el número de requisito dentro del tipo de requisito.

Ejemplo: **RF-1** Requisito Funcional, número 1

Se van a utilizar tablas como la siguiente:

IDENTIFICADOR	REQUISITO
Descripción	1.0 (Mayo 2010)
Prioridad	Alta, Media o Baja
Comentarios	Ejemplo

Tabla 11: Requisito de ejemplo

Requisitos funcionales:

RF-1	Conexión Tracker
Descripción	El usuario dispone de un servidor para la transferencia de archivos
Prioridad	Alta
Comentarios	Necesario para el funcionamiento. Se dispone del comando conectaTB

Tabla 12: RF-1: Conexión Tracker

RF-2	UPLOAD
Descripción	La aplicación necesita el registro del servicio de subida para que el usuario final pueda realizar sus funciones
Prioridad	Alta
Comentarios	Sin comentarios

Tabla 13: RF-2: UPLOAD

RF-2.1	**Crear .torrent**
Descripción	El usuario puede crear un archivo *.torrent* referenciado a un archivo y anunciarlo en el tracker
Prioridad	Alta
Comentarios	Se dispone del comando CT

Tabla 14: RF-1.2: Crear *.torrent*

RF-2.2	**Publicar .torrent**
Descripción	El usuario puede publicar en el tracker un archivo *.torrent* referenciado a un archivo
Prioridad	Alta
Comentarios	Se dispone del comando PT

Tabla 15: RF-2.2: Publicar *.torrent*

RF-2.3	**Compartir .torrent**
Descripción	El usuario puede publicar un archivo *.torrent* en el *tracker* para permitir la posterior descarga
Prioridad	Alta
Comentarios	Se dispone del comando PT

Tabla 16: RF-2.3: Compartir .torrent

RF-3	**DOWNLOAD**
Descripción	La aplicación necesita el registro del servicio de descarga para que el usuario final pueda realizar sus funciones
Prioridad	Alta
Comentarios	Sin comentarios

Tabla 17: RF-3: DOWNLOAD

RF-3.1	**Descargar archivo**
Descripción	El usuario puede descargarse un archivo disponible en el servidor, es decir, el *.torrent*, que lleva asociado el archivo, que previamente, haya sido anunciado, publicado y compartido en el *tracker*.
Prioridad	Alta
Comentarios	Se dispone del comando DT

Tabla 18: RF-3.1: Descargar archivo

Requisitos no funcionales:

RNF-1	**Equipos físicos**
Descripción	Los requisitos hardware que necesita. Un PC integrado ha sido seleccionado como la plataforma de hardware para RGW, ya que posee características similares a una puerta de enlace que supone recursos limitados. El PC está compuesto de un CPU Via C3 533MHz, 512MB de RAM, 2 interfaces Fast Etehernet,1Giga-bit Ethernet y una tarketa de 802.11b.La implementación OSGi ha sido proporcionada por la implementación Felix1.0(felix.apache.org). Todo el software ha sido instalado sobre la distribución Ubuntu 6.06 de Linux.
Prioridad	Alta
Comentarios	Ninguno

Tabla 19: RNF-1: Equipos físicos

RNF-2	**Comprobación del paradigma**
Descripción	Se comprobará que la aplicación se ha diseñado y desarrollado siguiendo la orientación a objetos de Java y la plataforma OSGi
Prioridad	Alta
Comentarios	Ninguno

Tabla 20: RNF-2: Comprobación del paradigma

RNF-3	Comprobación del protocolo
Descripción	Se comprobará que la aplicación cumple con el protocolo BitTorrent
Prioridad	Alta
Comentarios	Ninguno

Tabla 21: RNF-3: Comprobación del protocolo

RNF-4	Entorno Java
Descripción	La aplicación funcionará en cualquier máquina que tenga instalada la versión 6 del entorno JSE
Prioridad	Alta
Comentarios	Ninguno

Tabla 22: RNF-4: Entorno Java

RNF-5	Estructura lógica
Descripción	La aplicación sigue una estructura lógica para la subida de archivos, y otra para la descarga. Subida: Crear torrent-Publicar torrent-Compartir torrent. Descarga: Descargar torrent
Prioridad	Alta
Comentarios	La estructura puede tener variaciones pero se aconseja seguir el flujo normal del protocolo BitTorrent

Tabla 23: RNF-5 Estructura lógica

RNF-6	Comandos
Descripción	La aplicación dispone del registro de unos comandos necesarios para la utilización por parte del usuario.
Prioridad	Alta
Comentarios	

Tabla 24: RNF-6 Comandos

RNF-7	Rendimiento: Tiempo de descarga de un archivo
Descripción	El tiempo desde que un usuario ejecuta el comando Dt hasta que la descarga del archivo es completado.
Prioridad	Alta
Comentarios	Depende del tamaño del archivo y de la red en la que estamos trabajando

Tabla 25: RNF-7: Tiempo de descarga de un archivo

RNF-8	Documentación
Descripción	Información de la aplicación detallada
Prioridad	Alta
Comentarios	Incluye un manual de usuario con el mínimo nivel de complejidad para el fácil aprendizaje, entendimiento y manejo de la aplicación.

Tabla 26: RNF-8: Documentación

5.2 MODELO CONCEPTUAL

5.2.1 DIAGRAMA DE CLASES

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran la aplicación o sistema. Dicho diagrama está compuesto por las clases, con sus correspondientes, atributos, métodos o visibilidad, y las relaciones entre ellas, como puede ser herencia, implementación, asociación y uso.

A continuación, se muestra el diagrama de clases de la aplicación. Puesto que la aplicación completa, se divide en varias subaplicaciones, se mostrará la relación de cada aplicación por separado y finalmente un esquema completo. La figura 23, es la aplicación que implementa el Tracker o Servidor de la aplicación Torrent:

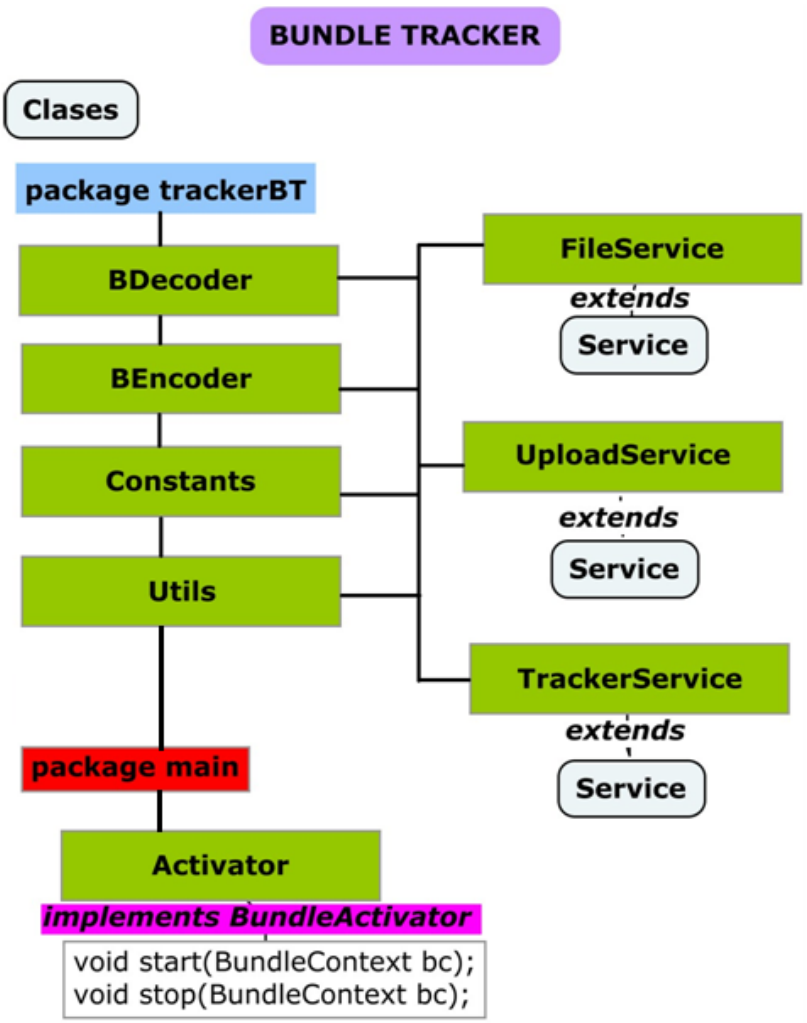

Figura 23: Diagrama de clases del BundleTracker

Las clases principales son FileService, esta clase procesa cualquier petición de un archivo, UploadService, servicio para cargar un archivo .torrent al tracker, TrackerService, controla la información de peer en el servidor, y la clase Activator del bundle que conecta el tracker y procesa las peticiones.

La aplicación que implementa el protocolo BitTorrent para que los clientes puedan descargar o subir ficheros, se corresponde con la figura 24.

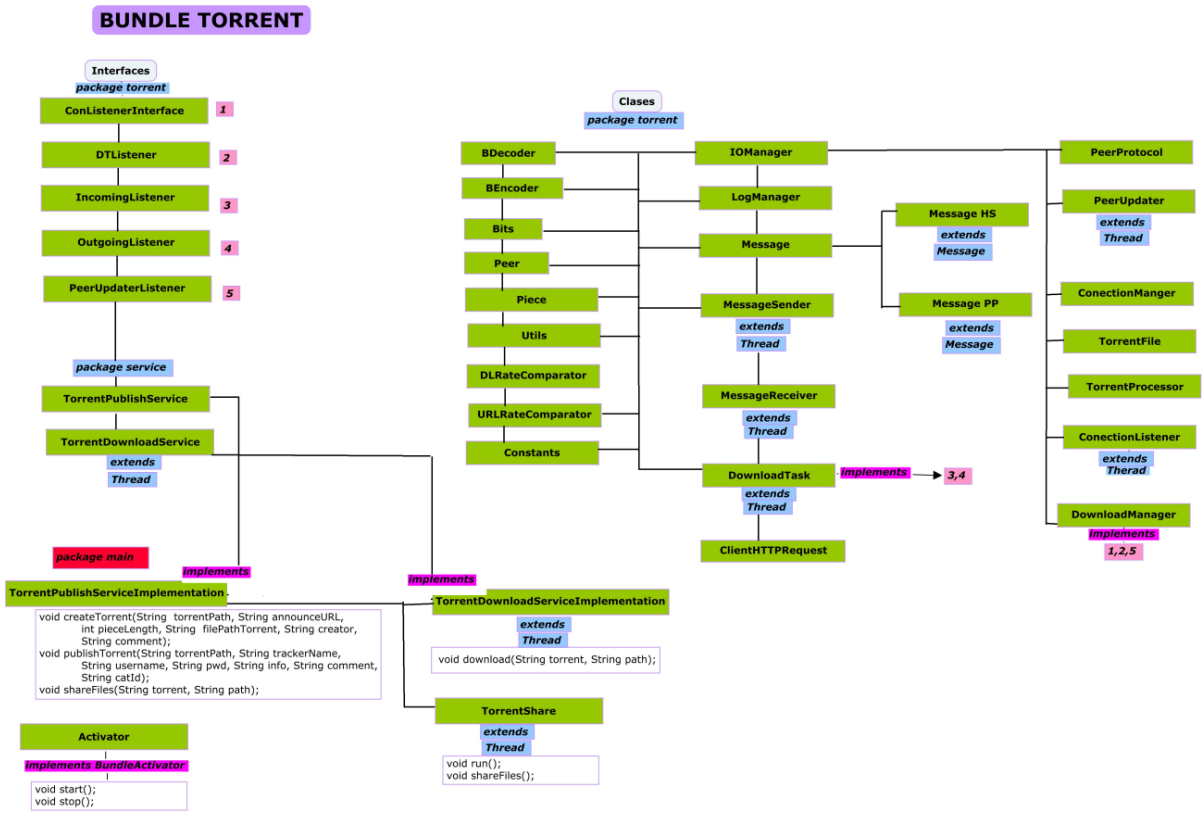

Figura 24: Diagrama de clases del BundleTorrent

Las clases principales de esta aplicación, se encuentran en los paquetes service y main. El paquete torrent, contiene las clases necesarias del protocolo y fueron descargadas de la página principal de BitTorrent.

El paquete service, contiene las interfaces de los servicios de subida y descarga de archivos, y las implementaciones se encuentran en el paquete main, TorrentPublishServiceImplementation y TorrentDownloadServiceImplementation. En el paquete main, se encuentra la clase Activator, y en su método start, son registrados estos dos servicios.

El diagrama de clases de la figura 25, es la aplicación que usa los servicios proporcionados por la aplicación Torrent, y proporciona el registro de una serie de comandos para permitir la

interacción con el usuario. En el método start del Activator del BundleClient, se registran los comandos comentados y se hace uso de los servicios TorrentPublishService y TorrentDownloadService.

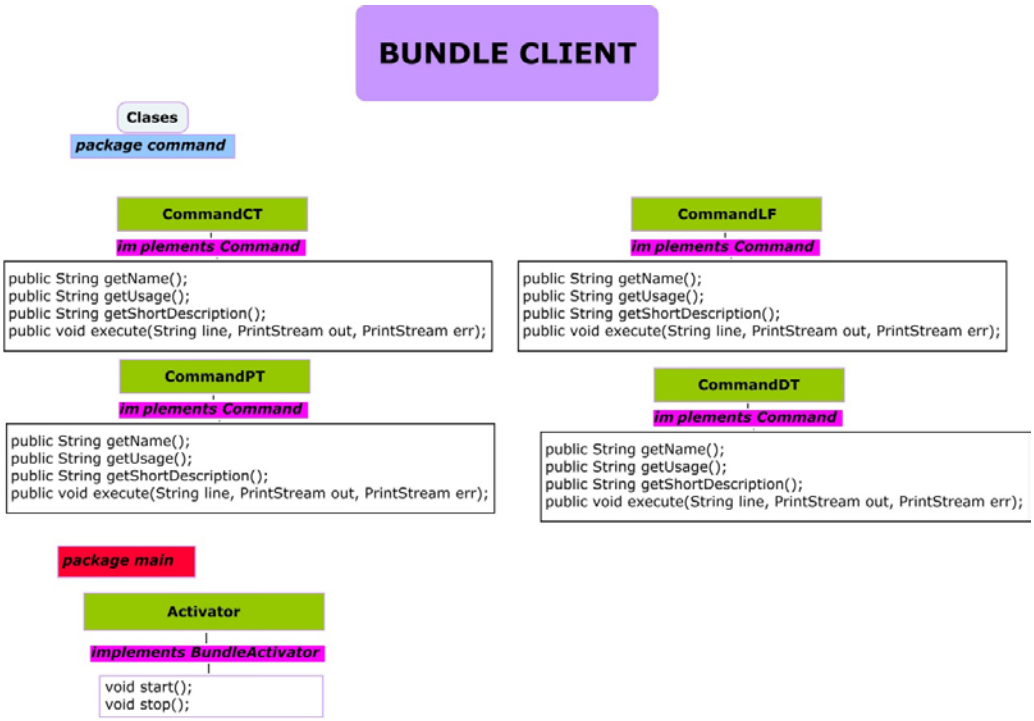

Figura 25: Diagrama de clases del BundleClient

Vistos los esquemas de los bundles, en la figura 26, se muestra un esquema general del proyecto completo.

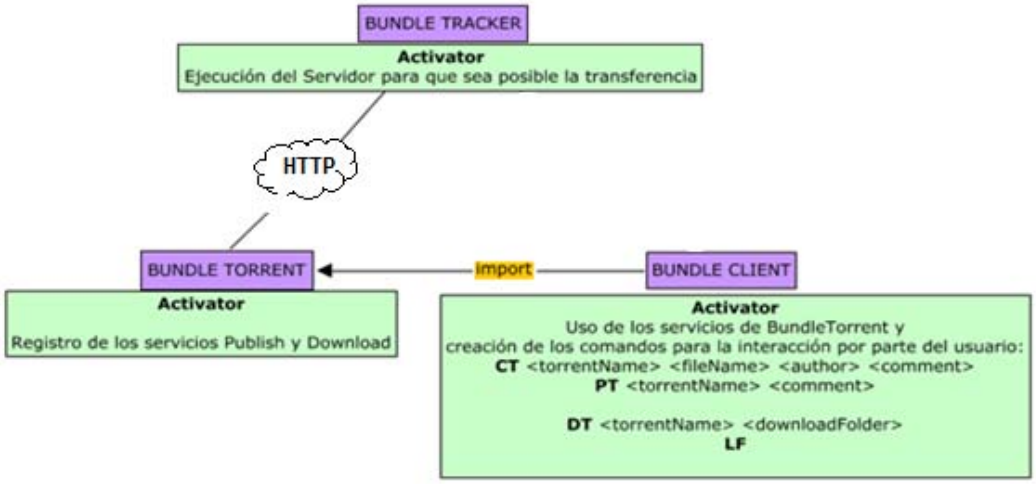

Figura 26: Diagrama General

Una vez mostrados los diagramas, se incluye un diagrama conceptual, donde se explica la función de cada una de las clases.

BUNDLE TRACKER

Jerarquía de Clases:

BUNDLE TRACKER	DESCRIPTION
Package trackerBT	
BDecoder	Contiene métodos de utilidad de conversión de parámetros
BEncoder	Contiene métodos de utilidad de conversión de parámetros
Constants	Constantes de utilidad
FileService	Servicio llamado para procesar cualquier petición de un archivo determinado
TrackerService	Servicio llamado para responder a un peer pidiendo la información de los *peers* que se están compartiendo en un torrent dado.
UploadService	Servicio llamado para cargar un archivo torrent al *tracker*
Utils	Métodos de utilidad usados por diferentes clases de la aplicación.
Package main	
Activator	Clase para la ejecución del *Bundle* y que procesa las peticiones de acuerdo al servicio dado. Se conecta mediante el protocolo HTTP

Tabla 27: Diagrama conceptual BundleTracker

BUNDLE TORRENT

Jerarquía de Interfaces:

BUNDLE TORRENT: INTERFACES	DESCRIPTION
Package torrent	
ConListenerInterface	Thread que escucha la conexión de *peers* en remoto para un cliente
DTListener	Maneja los eventos de la tarea de descarga
IncomingListener	Actúa cuando un mensaje está siendo recibido
OutgoingListener	Actúa cuando la conexión con el peer remota está siendo cerrada
PeerUpdaterListener	Interfaz para la actualización de *peers* disponibles
Package service	
TorrentPublishService	Métodos del servicio de publicación de archivos torrent
TorrentDownloadService	Métodos del servicio de descarga de archivos torrent

Jerarquía de Clases:

BUNDLE TORRENT	DESCRIPTION
Package torrent	
BDecoder	Contiene métodos de utilidad de conversión de parámetros
BEncoder	Contiene métodos de utilidad de conversión de parámetros
Bits	
ClientHTTPRequest	Clase para el envío HTTP
ConectionManager	Contiene métodos para la interacción de descarga y subida usando el protocolo HTTP
Constants	Constantes de utilidad
DLRateComparator	Compara la tasa de 2 *peers* descargados
DownloadManager	Manejador de las descargas
IOManager	Métodos para operaciones de Entrada/Salida
LogManager	Información de salida de un fichero
Message	Estructura general de un mensaje de un protocolo dado
Message HS	Mensaje con las especificaciones requeridas por el protocolo BitTorrent
MEssage PP	Mensaje con las especificaciones requeridas por el protocolo BitTorrent
Peer	Protocolo de compartición peer
PeerProtocol	Constantes usadas en el protocolo peer.
Piece	Representa cada "*piece*"(trozo) de acuerdo al protocolo BitTorrent
ConectionListener	Thread que escucha la conexión de *peers* en remoto para un cliente
DownloadTask	Representa la tarea de las "*pieces*" descargadas de un peer en remoto.
MessageReceiver	Thread creado para escuchar los mensajes de entrada de los *peers*.
MessageSender	Thread creado para enviar un mensaje al peer
PeerUpdater	Provee métodos para permitir la comunicación entre el cliente y el *Tracker*(Servidor)
TorrentFile	Representación de un fichero *.torrent*
TorrentProcessor	Procesador del torrent
URLRateComparator	Compara la tasa de 2 *peers* subidos
Utils	Métodos de utilidad
Package main	
Activator	Clase para la ejecución del Bundle y que registra el servicio del Torrent para que pueda interactuar con otros Bundles.
TorrentPublishServiceImplementation	Implementación del servicio de publicación de archivos *.torrent*
TorrentDownloadServiceImplementation	Implementación del servicio de descarga de archivos *.torrent*
TorrentShare	Clase para la compartición de archivos *.torrent*

Tabla 28: Diagrama conceptual del BundleTorrent

BUNDLE CLIENT

Jerarquía de Clases:

BUNDLE CLIENT	DESCRIPTION
Package commands	
CommandCT	Clase que crea el comando CT para que el usuario pueda crear un torrent.
CommandPT	Clase que crea el comando PT para que el usuario pueda publicar un torrent.
CommandDT	Clase que crea el comando DT para que el usuario pueda descargarse un torrent.
CommandFL	Clase que crea el comando FL, el cual mostrará una lista de los archivos disponibles en el directorio.
Package main	
Activator	Clase para la ejecución del *Bundle* y que registra el servicio de los comandos y usa a su vez el servicio Torrent proporcionado en la aplicación BundleTorrent.

Tabla 29: Diagrama conceptual del BundleClient

5.2.2 DIAGRAMA DE SECUENCIA

--- --- --- --- --- --- --- ---

Una vista dinámica que representa las interacciones de los objetos en un sistema. Estos diagramas, ilustran la interacción entre los objetos y el orden secuencial en el que ocurren dichas interacciones, es decir, como se comunican los objetos entre sí.

El primer mensaje en un diagrama siempre se inicia arriba en el lado izquierdo, y los demás van aumentando ligeramente hacia abajo. El mensaje, nombre del método, se coloca arriba de la flecha. El mensaje que se envía representa una operación o método que la clase objeto receptora va a implementar.

Las siguientes imágenes, muestran los diagramas de secuencia de la aplicación.

La figura 27, detalla cómo el usuario se conecta al tracker. Este paso será el primero a realizar en la ejecución de la aplicación. Este servidor permite la transferencia de archivos, además, lleva incluido un fichero de configuración XML, mediante el cual guarda información. Si el tracker no está activo, al querer compartir un torrent nos dará un error de que el tracker no está disponible.

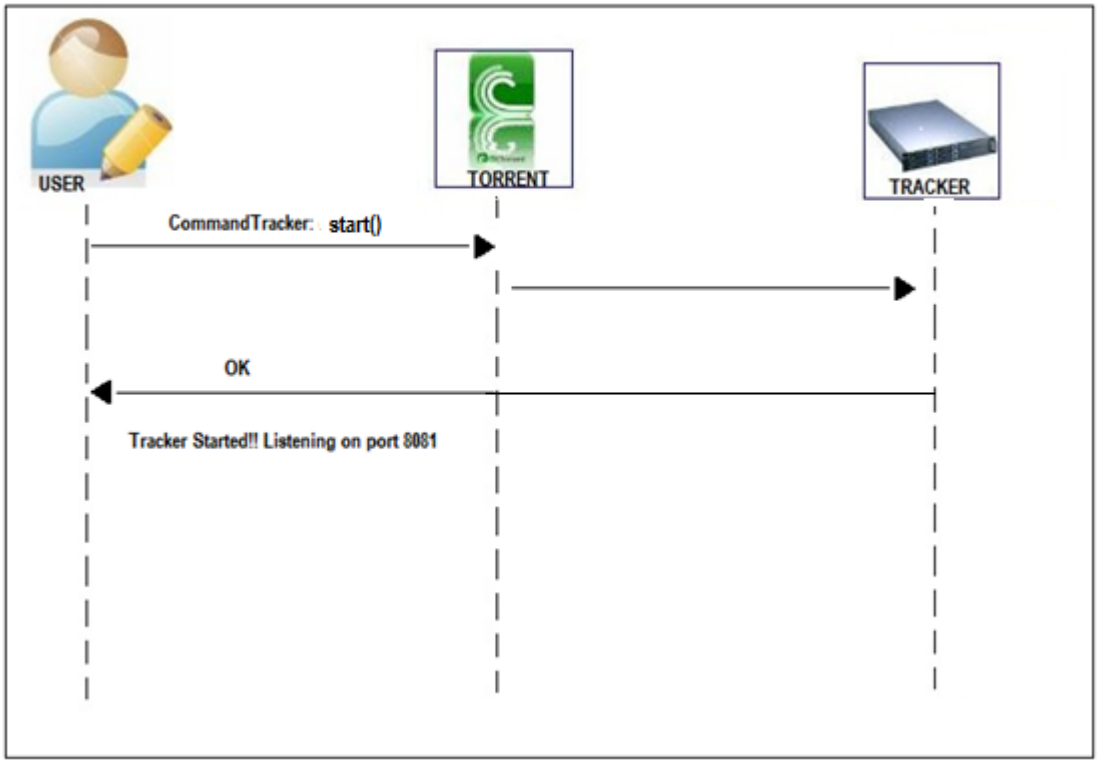

Figura 27:Diagrama de secuencias, CONEXIÓN TRACKER

En la siguiente ilustración, se incluye como se realiza la subida de archivos, y se explica la información que irá guardando el fichero de configuración del tracker.

En la figura 28, se muestra la subida de un torrent, la cual se realiza en dos pasos. Primero a través del comando CT se creará el torrent, al cual se le indica el tracker con el que va a establecer la comunicación; una vez creado habrá que publicarlo en el tracker, con el comando PT. Una vez publicado el tracker lo guarda en una base de datos, y posteriormente se comparte con este último comando y cualquier otro usuario podrá descargárselo.

El tracker provee estadísticas acerca del número de transferencias, el número de nodos con una copia completa del fichero (seeds) y el número de nodos que poseen sólo una porción del mismo (peers).

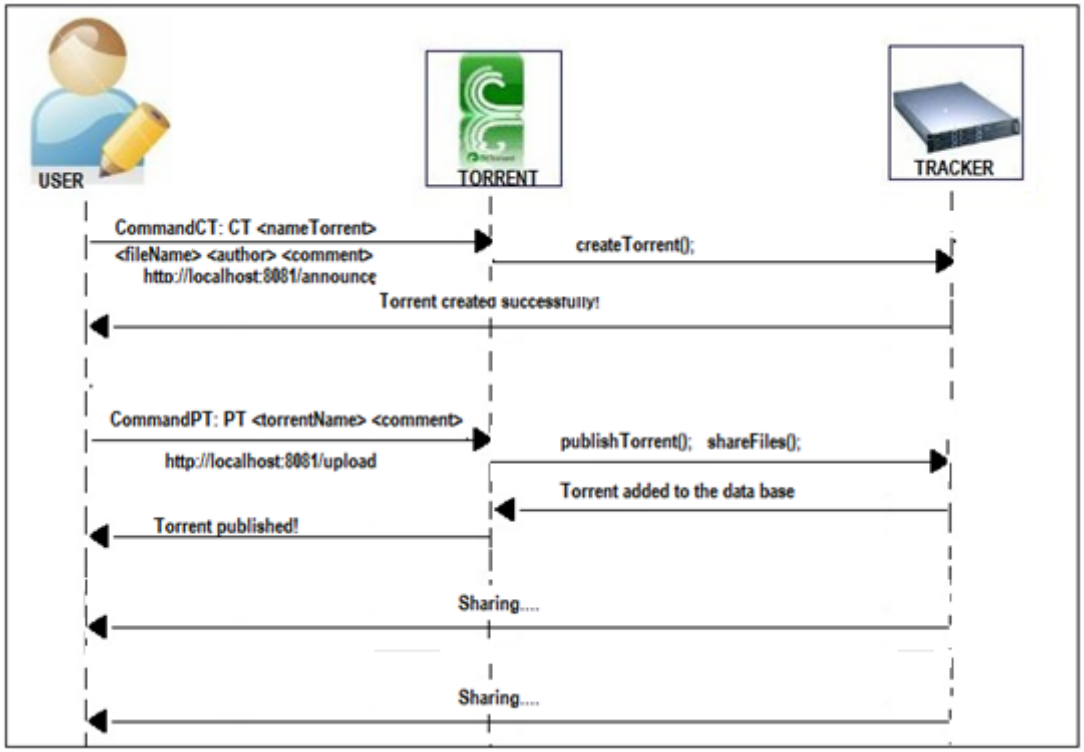

Figura 28: Diagrama de secuencias, UPLOAD

La estructura mostrada en la figura 27, es generalizada, pero pueden crearse varios torrents simultáneamente así como publicar uno y otro no o las diferentes opciones que se puedan dar.

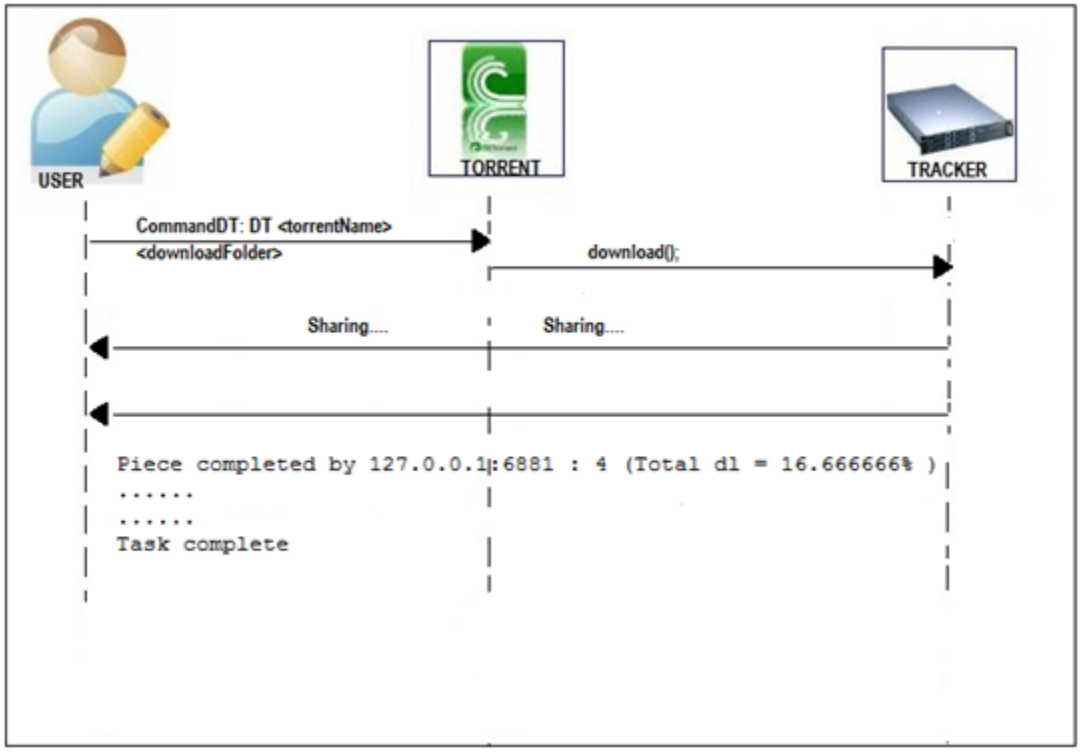

Figura 29: Diagrama de secuencias DOWNLOAD

La figura 29, muestra el proceso de descarga, el usuario únicamente ejecuta el comando DT con el nombre del torrent a descargar y esté irá siendo descargado en pieces hasta que aparece Task complete, que el torrent ya estará descargado en la carpeta indicada.

El tracker, a su vez, irá actualizando el número de peers disponibles, e irá almacenando la información en el archivo de configuración XML que tiene.

El siguiente diagrama que se presenta, es un ejemplo general que podría darse, para ello en el diagrama, vamos a suponer que el tracker ya ha sido activado por un cliente y que realiza sus funciones, de manera que se vea más clara la interacción de dos usuarios que usan la aplicación:

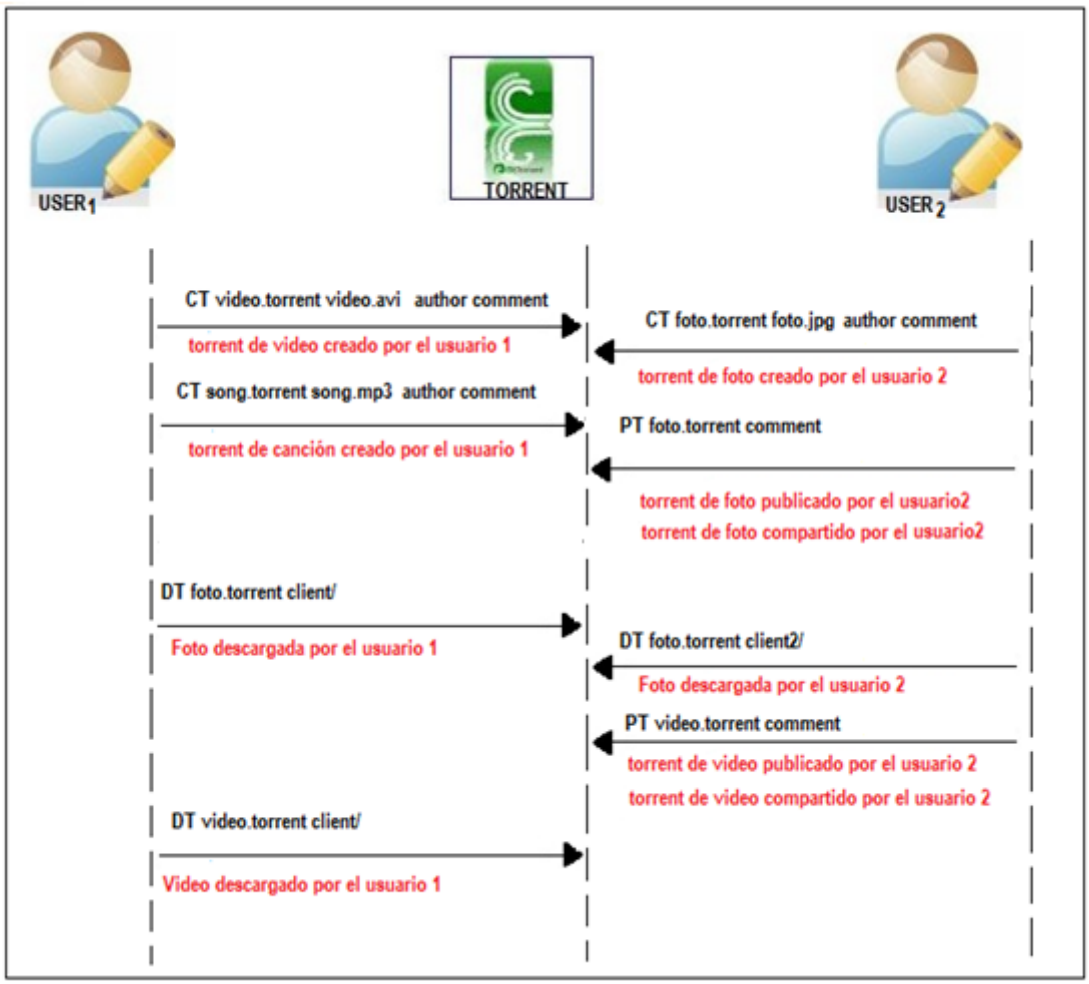

Figura 30: Diagrama de secuencias, GESTIÓN DE USUARIOS

Como se observa en el diagrama de la figura 30, dos clientes están interactuando a la vez. Como están conectados al mismo tracker, una vez creado un .torrent de un archivo existente en un determinado directorio, los pasos de publicación/compartición puede realizarlos el usuario1 o el usuario2. Los tres pasos requeridos en la parte de Upload, es muy importante

realizarlos, ya que si algún paso no se hace, cuando cualquiera de los dos usuarios que fuera a descargarse un archivo no lo recibiría correctamente. Realmente, el usuario realiza dos pasos, el de creación y publicación, aunque internamente, al publicar, también se realiza el paso de compartición.

Para poder descargarnos un archivo, tendrá que estar compartido, que se consigue mediante el comando PT y el usuario ya podrá descargarse el archivo directamente con el comando DT.

En el caso del ejemplo, la canción únicamente tiene creado su .torrent correspondiente, pero no podría ser descargada puesto que ha sido publicada ni compartida previamente.

6. IMPLEMENTACIÓN

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

En esta fase, se detalla la programación del sistema, es decir la realización detallada de la aplicación.

6.1 APLICACIÓN BundleTorrent

Esta aplicación, puede ser considerada el módulo principal de la aplicación completa, ya que implementa el protocolo BitTorrent, que es el servicio que queda registrado para posteriormente poder usado por los clientes que lo deseen.

En esta aplicación se incluyen tres paquetes que son definidos a continuación en detalle:

package torrent: este paquete contiene el código fuente del protocolo BitTorrent, proporcionado en [8]. Sus clases poseen métodos necesarios para la creación de los servicios definidos en el package service. Ha sido necesario realizar adaptaciones del código para poder integrarlo en OSGi.

package service: en este paquete se han definido dos interfaces con los métodos que serán necesarios implementar para que sea posible la transferencia de archivos. La interfaz TorrentPublishService, mostrada en la tabla 30, contiene los métodos para realizar la parte de upload y la interfaz TorrentDownloadService, tabla 31, el método para realizar el download.

```
package service;

public interface TorrentPublishService {

    public void createTorrent(String  torrentPath, String announceURL,
            int pieceLength, String  filePathTorrent, String creator,
            String comment);
    public void publishTorrent(String torrentPath, String trackerName,
            String username, String pwd, String info, String comment,
            String catId);
    public void shareFiles(String torrent, String path);

}
```

Tabla 30: Interfaz de subida de archivos

TorrentPublishService.java: El método createTorrent, lo que hace es generar un archivo con la extensión .torrent. La información que se incluye es la dirección del servidor dónde se va a compartir, el nombre y comentario indicado por el usuario, el número de trozos en los que el archivo original será descargado y un código de hash que hace referencia al archivo original que es el que en definitiva, será descargado por algún cliente.

Una vez creado, el método publishTorrent, lo que hace es publicarlo en el tracker para que posteriormente pueda ser compartido. Esta compartición, se realiza a través del shareFiles, que compartirá el torrent indicado, en el directorio también indicado.

```
package service;


public interface TorrentDownloadService {

    public void download(String torrent, String path);

}
```

Tabla 31: Interfaz de descarga de archivos

TorrentDownloadService.java: únicamente posee el método download, que lo que hace es descargar el torrent que le pasa como parámetro en el directorio que indicamos.

package main: se incluyen las implementaciones de los servicios del paquete service(TorrentPublishServiceImplementation y TorrentDownloadServiceImplementation). Es decir, se lleva a cabo la implementación de los métodos mostrados en las dos ilustraciones superiores. Además se incluye la clase TorrentShare, que es un Thread que permite la compartición de varios archivos sin que el proceso de transferencia de un archivo afecte a otro que está siendo transferido en ese momento. Por último, como ya se ha comentado en el apartado 3.3.2 Bundles, se necesita de la clase Activator que debe incluir los métodos start() y stop() para arrancar o parar la aplicación. Como se muestra en la tabla 32, en el método start(), se han registrado los dos servicios implementados para que puedan usarse por otras aplicaciones o bundles.

```
    TorrentDownloadService ds;
    TorrentPublishService us;
    BundleContext bc;

// Registro el servicio de subida de archivos
us = new TorrentPublishServiceImplementation();
bc.registerService(TorrentPublishService.class.getName(), us, null);

System.out.println("Servicio de subida registrado");


// Registro el servicio de descarga de archivos
ds = new TorrentDownloadServiceImplementation();
bc.registerService(TorrentDownloadService.class.getName(), ds, null);

System.out.println("Servicio de bajada registrado");
```

Tabla 32: Registro de los servicios

6.2 APLICACIÓN BundleTracker

El servidor, estaba también proporcionado en [8], por lo que únicamente ha habido que adaptar el main de este programa para poderlo incluir en el start() de nuestro bundle, en resumen, para integrarlo en OSGi.

package trackerBT: contiene todas las clases necesarias para el funcionamiento del servidor. Durante el desarrollo del proyecto estas clases no han sido modificadas.

package main: en este paquete se incluye la clase Activator para la ejecución del tracker. El método start() contiene el código necesario que se ponga a escuchar. Además, necita un fichero XML de configuración, tabla 33, el cual se obtiene a través del método getProperty del bundle, con la ventaja de que si el día de mañana necesitáramos cambiar el fichero, se haría en el fichero de configuración de Felix sin necesidad de tener que tocar código.

```
String ficheroTracker=bc.getProperty("BundleTracker.ficheroTracker");
```

Fichero de configuración:

```
<?xml version="1.0" ?>
<config>
  <param name="context" value="./example/tracker/" />
  <param name="torrentXML" value="./example/tracker/files/torrents.xml" />
  <param name="peerXML" value="./example/tracker/files/peers.xml" />
  <param name="listeningPort" value="8081" />
  <param name="servername" value="JBitTorrentAPI Tracker" />
  <param name="torrentsDir" value="./example/tracker/files/torrentsFile/" />
</config>
```

Tabla 33: Fichero de configuración del Tracker

6.3 APLICACIÓN BundleClient

Esta aplicación es la que hace que un cliente o usuario del servicio BundleTorrent, pueda interactuar, es decir usa los servicios TorrentPublishService y TorrentDownloadService.

El objetivo de esta aplicación ha sido permitir al usuario un manejo sencillo, dado que el presente proyecto no va a disponer de una interfaz gráfica. Para ello, se han creado unos

sencillos comandos que son registrados en la consola cmd, y ejecutando cada uno de ellos, se consigue el uso completo del programa.

package command: se incluye una clase por cada comando que se ha definido. Estas clases implementan la interfaz command, tabla 34, incluida en el paquete org.apache.felix.shell.Command.

```
public interface Command{

public String getName();
public String getUsage();
public String getShortDescription();
public void execute(String s, PrintStream out, PrintStream err);
}
```

Tabla 34: Interfaz command

A continuación, se describe cada una de las clases del paquete command.

CommandCT.java: comando que sirve para crear el .torrent. En el método *execute* se hace una llamada al *createTorrent* de *TorrentPublishService*, y la información que se pide al usuario es el nombre del torrent, el nombre del archivo, el autor y el comentario. El resto de parámetros necesarios para hacer la llamada al método, así como los directorios, o la dirección del tracker son incluidos en el fichero de configuración de Felix, y se obtienen a partir del método *getProperty* del *bundle*.

```
String defaultLocation = bc.getProperty("BundleClient.defaultLocation");
String urlTracker = bc.getProperty("BundleClient.urlTracker");
ArrayList<String> parametros= new ArrayList<String>();

tps.createTorrent(
        defaultLocation + parametros.get(1),
        urlTracker + "announce",
        256,
        defaultLocation + parametros.get(2),
        parametros.get(3),
        parametros.get(4));
```

Tabla 35: Método *execute* del comando CT

CommandPT.java: comando que sirve para publicar el torrent que se le indica, con un comentario. Al igual que en la clase *CommandCT*, el resto de parámetros necesarios para hacer la llamada al método, así como los directorios, son incluidos en

el fichero de configuración de Felix, y se obtienen a partir del método *getProperty* del *bundle*. En el método *execute* se hace una llamada al método *publishTorrent* de la clase *TorrentPublishService*. Además, sirve para compartir el torrrent idicado, haciendo una llamada al método *shareFiles* de la clase *TorrentPublishService*.

```
String defaultLocation = bc.getProperty("BundleClient.defaultLocation");
String urlTracker = bc.getProperty("BundleClient.urlTracker");
String user=bc.getProperty("BundleClient.user");
String psw=bc.getProperty("BundleClient.psw");
String info=bc.getProperty("info");
String id=bc.getProperty("id");

tps.publishTorrent(
        defaultLocation + parametros.get(1),
        urlTracker + "upload",
        user,
        psw,
        info,
        parametros.get(2),id);
tps.shareFiles(defaultLocation + parametros.get(1),defaultLocation);
```

Tabla 36: Método *execute* del comando PT

CommandDT.java: comando que sirve para descargar el torrent que se le indica, y el nombre de la carpeta donde va a ser descargado. Es necesario el directorio donde se va a descargar el archivo, que está incluido en el fichero de configuración de Felix, y se obtiene a partir del método *getProperty* del *bundle*. En el método *execute* se hace una llamada al método *download* de la clase *TorrentDownloadService*.

```
String defaultLocation = bc.getProperty("BundleClient.defaultLocation");
String downloadLocation=bc.getProperty("BundleClient.downloadLocation");

tds.download(defaultLocation + parametros.get(1),downloadLocation + parametros.get(2));
```

Tabla 37: Método *execute* del comando DT

CommandLF.java: lo que hace es mostrar una lista de los archivos disponibles en el directorio indicado en el fichero de configuración de Felix. El directorio está definido en la variable *defaultLocation*

```
public void execute(String s, PrintStream out, PrintStream err){
    String[]ficheros;
      File directorio=new File("C:/TORRENT/UPLOAD/");

      ficheros=directorio.list();

      if(ficheros==null){
          System.out.println("No hay ficheros disponibles en este directorio");
      }else{
          for(int i=0;i<ficheros.length;i++){

              System.out.println(ficheros[i]);


          }

    }
```

Tabla 38: Método execute del comando LF

La figura 31, muestra como sería el uso de cada uno de estos comandos en la cmd cuando introducimos el comando help.

```
help
CT <torrentName> <fileName> <author> <comment> - Creates the .torrent.
DT <torrentName> <downloadFolder>              - Downloads the .torrent indicate
d.The download folder can be client1 or client2
LF                                             - Shows a list with all the avail
able files in the folder UPLOAD
PT <torrentName> <comment>                     - Publish the .torrent indicated.
```

Figura 31: Ayuda de los comandos

Nota: La dirección del tracker apunta al localhost, es decir, en principio está configurado para trabajar con un único PC, en local. En caso de querer trabajar más ordenadores, bastaría con indicar la dirección IP correspondiente en la variable UrlTracker del fichero config.properties [12].

6.4 DECISIONES DE DISEÑO

--- --- --- --- --- --- --- ---

Con las decisiones tomadas en el diseño de la implementación, se pretende el mejor uso de la aplicación.

6.4.1 GESTIÓN UPLOAD

--- --- --- --- --- --- --- ---

Creación del archivo .torrent

Dado que no se va a disponer de interfaz gráfica, la manera más cómoda de crearlo era a través de un comando. Por ello se ha implementado el comando CT. En la creación, es

necesaria cierta información, que podemos ahorrar al usuario, como la ruta del directorio, la dirección del tracker… por lo que se ha decidido definir algunos de estos parámetros en el fichero de configuración de Felix como ya se ha comentado en el apartado 6.3. En conclusión, con indicar el nombre del torrent que se quiere crear, el archivo del cual se quiere crear el torrent, el autor y el comentario, bastará por parte del usuario.

Publicación/Compartición del archivo .torrent

Para la publicación del archivo, se ha decidido en el presente proyecto dejar ya por defecto la URL del tracker, de manera que el usuario no tenga que introducirla en el comando. Además, a la hora de publicarlo, está la opción de introducir un usuario y contraseña, los cuales se ha decidido poner defecto de tal manera que no sean necesarios. Cualquier implementación posterior sería configurable desde Felix sin tener que tocar el BundleClient. Por comodidad para el usuario, con indicar el nombre del torrent a publicar y un comentario bastará.

Por último, en la compartición, con indicar el nombre del torrent a compartir bastará. Según el API de BitTorrent, es necesario también indicar la ruta donde se encuentra ese torrent, pero la implementación se ha hecho de tal manera que no haga falta ninguno de estos parámetros y coja esta información del archivo de config.properties de Felix.

La gestión de upload/subida, podría haberse optimizado en un solo método, y por tanto un solo comando el cual implementara los tres pasos de creación, publicación y compartición, pero dado el funcionamiento del protocolo BitTorrent, se ha decidido dejar en funcionamiento las dos posibilidades por separado, ya que por ejemplo, un usuario puede querer crear un torrent pero no querer publicarlo ni compartirlo en principio.

6.4.2 GESTIÓN DOWNLOAD

--- --- --- --- --- --- --- ---

La gestión de descarga, es bastante simple, basta con indicar el torrent correspondiente al archivo que queremos descargar y la carpeta donde queremos que se descargue. Por defecto, se descargará en el directorio indicado en el fichero config.properties, definido en la variable downloadLocation y en la carpeta que indique el usuario.

7. PRUEBAS

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

Esta fase, nos permitirá determinar el nivel de calidad y el grado de cumplimiento con respecto a las especificaciones del sistema.

Se ha estado trabajando en remoto durante todo el proceso de creación del servicio BitTorrent, por lo que se ha trabajado únicamente con un PC, con sistema operativo Windows 7 , con un procesador intel core duoII T6600 y 4GB de memoria RAM. En este sistema, se ha instalado la plataforma Felix y en dicha plataforma, el servicio BitTorrent programado, con las tres aplicaciones o bundles comentados.

En la figura 32, se muestra como queda el entorno de trabajo de manera gráfica.

Figura 32: Entorno de trabajo

7.1 PRUEBAS DE INTEGRACIÓN

--- --- --- --- --- --- --- ---

Las pruebas de integración, se han realizado para comprobar la conexión del servidor o tracker. Este servidor implementa el protocolo HTTP, que está diseñado para transferir hipertextos, páginas web o páginas HTML. Posee un archivo de configuración XML, del que recoge la información del puerto en el que va a escuchar y además, con este archivo irá registrando si el torrent ya está en la base de datos del servidor, y la información de sedes y peers.

PRUEBA	RESULTADO	INFORMACIÓN ADICIONAL
Conexión Tracker	OK	Tracker listening on port 8081

Tabla 39: Pruebas de integración

7.2 PRUEBAS DE FUNCIONAMIENTO

Para comprobar el funcionamiento de la aplicación BitTorrent, se han realizado diferentes pruebas. A su vez, se han dividido en dos fases, la primera, la comprobación del buen funcionamiento de las pruebas que pueden realizarse individualmente y a continuación la segunda fase de pruebas conjuntas.

Fase I: Pruebas individuales:

PRUEBA	**RESULTADO**	**INFORMACIÓN ADICIONAL**
Conexión Tracker	OK	*Tracker listening on port 8081*
Funcionamiento Torrent		
Servicio UPLOAD	OK	*Servicio de subida registrado listo para ser usado por el cliente*
Servicio DOWNLOAD	OK	*Servicio de descarga registrado listo para ser usado por el cliente*

Tabla 40: Pruebas individuales

Fase II: Pruebas conjuntas

Funcionamiento del Tracker cuando hay transferencia de archivos: interacción del Tracker.jar con el Torrent.jar y el Client.jar.

PRUEBA	RESULTADO	INFORMACIÓN ADICIONAL
Conexión Tracker	OK	*El tracker comientza a escuchar:* *Tracker listening on port 8081*
Torrent añadido a la base de datos	OK	*Cuando se crea un torrent se añade a la base de datos del servidor*
Detección de torrents en la base de datos del servidor	OK	*Cuando se crea un torrent, que ya estaba creado, nos avisa de que ya está en la base de datos*
Actualización de las lista de peers desde el tracker	OK	*Cuando se termina la subida del archivo se actualista la lista de peers disponibles*

Tabla 41: Funcionamiento del Tracker

Funcionamiento del Torrent: esta aplicación conlleva pruebas individuales, y es usada por la aplicación Client, por lo que en esta fase de pruebas conjuntas se comprobará el correcto funcionamiento del Client que a su vez nos determinará el correcto funcionamiento del Torrent.

Funcionamiento del Client, interacción del usuario para realizar la transferencia de archivos. Interacción de Client.jar con el Torrent.jar y el Tracker.jar.

PRUEBA	RESULTADO	INFORMACIÓN ADICIONAL
Comando LF	OK	*Muestra la lista de archivos disponibles en el directorio C:/TORRENT/UPLOAD*
Comando CT	OK	*Crea el torrent referenciado al archivo indicado y lo anuncia al tracker*
Command PT	OK	*Publica el torrent indicado en el tracker y lo comparte para que pueda ser descargado*
Commnand DT	OK	*Descarga el archivo que contiene el torrent indicado en la carpeta que se le indica del directorio C:/TORRENT/DOWNLOAD*

Tabla 42: Funcionamiento del cliente

Fase III: Pruebas de tiempo de descarga

Una vez probado el funcionamiento de la aplicación, y que los resultados han sido OK, pasamos a probar la transferencia de archivos más pesados para hacernos una idea del tiempo de transferencia en función del tamaño del fichero. Los tiempos reflejados en las tablas 43,44 y 45, se han obtenido haciendo las pruebas con un solo ordenador en local.

PRUEBA	RESULTADO	INFORMACIÓN ADICIONAL
Carga de un fichero de tamaño: 2 a 5 MB aprox.	OK	*En cuestión de ms el archivo queda cargado y listo para ser descargado en el tracker. Tamaño del .torrent: 1KB*
Carga de un fichero de tamaño: 50 a 100 MB aprox.	OK	*La carga se realiza correctamente en un tiempo aproximado de 1 segundo y medio.Tamaño del .torrent: 9KB*
Carga de un fichero de tamaño: 800 MB	OK	*La carga de este fichero, que ya es más pesado se realiza en unos 14 segundos. Tamaño del .torrent: 62KB*

Tabla 43: Tiempos de carga de ficheros

PRUEBA	RESULTADO	INFORMACIÓN ADICIONAL
Descarga de un fichero de tamaño: 2 a 5 MB aprox.	OK	*El tiempo que tarde aproximadamente en descargar archivos de estos tamaños es de unos 5 segundos.*
Descarga de un fichero de tamaño: 50 a 100 MB aprox.	OK	*En ficheros de tamaño medio, el tiempo de descarga asciende a unos 27 segundos*
Descarga de un fichero de tamaño: 800 MB	OK	*La descarga en un archivo pesado de alrededor de 800 MB es de 2 minutos y 33 segundos aproximadamente.*

Tabla 44: Tiempos de descarga de ficheros

PRUEBA	RESULTADO	INFORMACIÓN ADICIONAL
Descarga simultanea de varios ficheros	OK	*Los tiempos de las dos tablas anteriores ascienden si un usuario está compartiendo a la vez varios ficheros.*
Descarga por parte de dos usuarios de un archivo de 800;B	OK	*En este caso, el tiempo de descarga asciende a 4 minutos y 10 segundos, pero en ese tiempo los dos usuarios tendrían descargado el archivo.*

Tabla 45: Descargas simultáneas

Nota: estos tiempos varían en función de la red usada, del número de archivos y del número de usuarios que están interactuando.

Pruebas con 2 PC´s:

Las últimas pruebas que se han realizado, ha sido usar dos ordenadores. Un ordenador actuaba como cliente UPLOAD, creando el torrent, publicándolo y compartiéndolo, y el otro ordenador actuaba como cliente DOWNLOAD, descargándose los archivos compartidos por el otro PC.

Los tiempos de descarga de archivos menores a 100MB, han sido aproximadamente los mismos, y el tiempo de descarga del fichero de 800MB ha ascendido a unos 10 minutos. Estos tiempos varían en función de la red.

8. CONCLUSIONES

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

Llegados a este capítulo de conclusiones, el objetivo principal de crear un servicio para plataformas OSGi, se da por completado. Se dispone de la posibilidad de compartir archivos usando los servicios proporcionados por la aplicación Torrent. El cliente puede interactuar con la aplicación mediante los comandos implementados.

El OBJETIVO1, consistía en la integración del protocolo BitTorrent en la plataforma de servicios OSGi. Para conseguirlo, se realizó un estudio a fondo del protocolo BitTorrent, como realiza la transferencia, cuales son los pasos a seguir, y el tipo de red que emplea para que la transmisión sea los más rápida y efectiva posible. También, se estudió en profundidad la plataforma OSGi, puesto que es necesario su conocimiento para el desarrollo de las aplicaciones.

Estudiado el funcionamiento, se procedió a la descarga del API de BitTorrent. En él se incluye, la funcionalidad del protocolo en sí, y el servidor que se va a utilizar para la transferencia de ficheros. Se realizó un análisis exhaustivo de todas ellas para realizar la adaptación necesaria con el fin de poder integrarlo en la plataforma OSGi.

Vista la estructura de clases, se optó por la creación de dos servicios paralelos, en lo referente al protocolo, el servicio de subida o *upload*, y el de descarga o *download*. Para cumplir con las diferentes funciones del protocolo BitTorrent (crear el torrent, publicarlo, compartirlo y descargarlo), en cada uno de los servicios se decidió crear los métodos necesarios, que implementaran cada una de estas funciones. La parte de adaptación del protocolo, ha dado bastantes problemas, ya que en los modificaciones de código, había funcionalidades que se perdían. Un ejemplo, es cuando para la parte de subida, se crearon los métodos *createTorrent*, *publishTorrent* y *shareFiles*, que cuando se invocaba el método de compartición esta no se realizaba correctamente y el usuario que hacía la parte de descarga, cuando iba a abrir el fichero, era erróneo y no podía ser abierto. Estos problemas fueron resueltos cumpliendo con los OBJETIVOS 2 y 3.

Referente al servidor o *tracker*, ya poseía un *main* con la ejecución que servía directamente para la integración, la única modificación necesaria era la lectura del fichero de configuración que necesita para ser ejecutado, que se indicaría en las propiedades del archivo de configuración de Felix. Con esta integración se cumple el OBJETIVO 5 del capítulo 2.

A día de hoy tenemos nuestra aplicación Tracker y nuestra aplicación Torrent, que implementa los servicios de upload (*TorrentPublishService*) y download (*TorrentDownloadService*).

Para el OBJETIVO4 de completar el proyecto con la creación de una aplicación que funcione demostrando la funcionalidad proporcionada por BitTorrent, se diseñó la aplicación Client. Esta aplicación interactúa con el *bundle* Torrent, y usa los servicios de dicho *bundle*. Con esta aplicación, el usuario podrá realizar todas las funcionalidades. Para que el usuario pueda manejarse por la aplicación, se han registrado una serie de comandos que permitan trabajar cumpliendo todos los objetivos de implementación del protocolo.

Como conclusiones haciendo referencia a la plataforma de servicios OSGi, destacar que este proyecto me ha ayudado a la comprensión de esta tecnología, la cual conocí hace unos años pero solo por encima.

Al igual, con el protocolo BitTorrent, el cual había utilizado en alguna ocasión pero desconocía su funcionamiento en un nivel más técnico.

Gracias a la ventaja que posee esta tecnología en cuanto a la modularidad, para futuras implementaciones bastaría con crear nuevas aplicaciones de clientes, pero el módulo del Torrent y el Tracker podrían seguir dando soporte a otros nuevos módulos de uso de los servicios anteriores.

Como conclusión a nivel personal, este proyecto me ha ayudado a ponerme al día en protocolos de transferencia, aunque es un mundo muy amplio, en el que hay gran cantidad de diferentes protocolos, que a su vez poseen diferentes clientes.

Además, aunque este proyecto no sea de domótica, sí que se habla de ella en la introducción a la pasarela residencial para justificar las necesidades de la pasarela y me gustaría comentar que me parece un tema muy actual y con muchas posibilidades de desarrollo para el futuro, y entre estas posibilidades, muchas de ellas serán para ayudar a personas con discapacidades, por lo que todavía quedan muchas ramas abiertas y muchas posibilidades en el mundo de la domótica.

9. LÍNEAS FUTURAS DE DESARROLLO

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

Nuestra aplicación, podría incorporar más funcionalidades y mejoras en un futuro, las cuales la harían más atractiva.

Para facilitar el manejo por el usuario, sería muy interesante la creación de una interfaz gráfica. Los clientes al ejecutar la aplicación, dispondrían de una pantalla con un menú con las instrucciones oportunas, así como la opción de realizar la transferencia de archivos seleccionando, a través de un explorador, el archivo a subir y en el caso de la descarga, incluir en la aplicación un buscador de archivos, que al seleccionarlo le indicáramos el directorio de nuestro PC dónde queremos descargarlo y acto seguido comience a descargarse. Sería una opción que tendría un atractivo visual para el usuario.

De esta manera, complementaríamos la implementación de los comandos. Una vez más, esta modificación solo afectaría al módulo del cliente, por lo que el resto de aplicaciones seguirían funcionando sin necesidad de ser tocadas, en lo que a programación de código se refiere.

Esta sería una mejora a nivel de adquisición del producto, dada la comodidad, y que siempre una buena interfaz visual atrae clientes.

Sería posible el registro de más comandos para ampliar funcionalidades. Uno de estos comandos podría ser, que en el caso de que un cliente tenga claro que quiere publicar un archivo para compartirlo desde ese momento, este proceso pueda hacerse en un único paso. Además, otras aplicaciones gráficas podrían usar los servicios proporcionados de manera automática y transparente al usuario, lo que supondría un uso diferente de estos servicios y también viable.

Según el protocolo BitTorrent, cuando un archivo *.torrent* es publicado en el *tracker*, pueden incluir un usuario y una contraseña para mayor seguridad y que no puedan ser descargados por un cliente no autorizado. Por defecto, hemos indicado que no habrá ni usuario ni contraseña, pero para un uso futuro, podría incluirse en la publicación del torrent esta información si se requiere.

Como ya se ha comentado, diferentes aplicaciones podrían hacer uso de este servicio. Una de ellas, sería el caso de una aplicación de audio bajo demanda, en la que un usuario recibiera mensualmente una actualización con las nuevas canciones que han salido al mercado durante el mes y pueda descargarse las que desee, a través del protocolo BitTorrent.

Otra aplicación, sería la creación de una página web donde se publiquen archivos *.torrent* y que alberga una aplicación para la descarga inmediata de estos archivos. A su vez, podría albergar una aplicación en la que cualquier usuario pueda publicar un archivo únicamente seleccionando el archivo desde su PC. La creación del torrent y su posterior publicación y compartición se haría de forma automática y no afecta al usuario.

El servicio de BitTorrent para plataformas de servicios OSGi, ofrece posibilidades a muchas aplicaciones multimedia y de entretenimiento.

BIBLIOGRAFÍA

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

Lo más importante para comenzar a desarrollar un proyecto, es contar con la información necesaria. Para ello, se ha consultado gran información en Internet, en documentos y en proyectos estabas relacionados con alguna integración de aplicaciones en OSGi.

Osgi/Domótica

[1] http://osgimetrics.blogspot.com/2009/07/creacion-de-paquetes-con-servicio-y-uso.html

[2] http://www.ub.es/geocrit/sn/sn-146%28136%29.htm

[3] http://en.wikipedia.org/wiki/OSGi

[4] http://www.domoticaviva.com/noticias/016-100802/pasarela4.htm

[5]http://www.discapnet.es/Castellano/areastematicas/Accesibilidad/Accesibilidadenelhogar/Domoticaydiscapacidad/Documents/Guias/Domotica/index.html

[6] http://www.domoprac.com/domoteca/23/257-historia-de-la-domotica-pasado-presente-y-futuro.

[7] http://www2.udec.cl/~carlosalvial/domotica/pages/que_es.htm

BitTorrent

[8] http://sourceforge.net/projects/bitEx

[9] http://www.bittorrent.com/

[10] http://www.ayudabittorrent.com/funcionamiento-protocolo-bittorrent

[11] http://www.maestrosdelweb.com/editorial/bittorrent/

[12] http://www.ignside.net/man/servidores/host.php

[13] http://temporalmente.wordpress.com/2009/03/07/tutorial-bittorrent-para-principiantes/

Java

[14] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java

[15] http://es.tech-faq.com/java.shtml

Apache-Felix

[16] http://felix.apache.org/site/index.html

[17] http://felix.apache.org/site/apache-felix-osgi-tutorial.html

[18] Ejecución de los bundles:

http://www.oracle.com/technology/pub/articles/dev2arch/2007/12/osgi-introduction3.html?_template=/ocom/print

[19] Creación de comandos: http://felix.apache.org/site/apache-felix-shell.html

Eclipse- Knopflerfish

[20] http://felix.apache.org/site/integrating-felix-with-eclipse.html

[21] http://www.knopflerfish.org/eclipse_plugin.html

[22] http://mikiorbe.wordpress.com/2009/02/23/desarrollo-osgi-en-eclipse/

[23] http://eclipsetutorial.forge.os4os.org/in1.htm

Documentos/Proyectos

[24] Ideas para la definición de una plataforma universal. José Manuel Márquez Vázquez. Lenguajes y sistemas informáticos. Universidad de Sevilla.

[25] Apéndice A: OSGi: Open Services Gateway Interface. Dr.Diego de Artaza. Universidad de Deusto

[26] Estudio y evaluación de la especificación OSGi para arquitecturas móviles. Jaime Alberto Flautero Valencia y Fabio Andrés Sánchez Baptiste. Ingeniería de sistemas

[27] Servicios de transferencia de ficheros para la plataforma OSGi. PFC de Daniel Acedos Casado

[28] Sistema de gestión de citas médicas en entornos de Tele-asistencia y Tele-seguimiento. PFC de Adrián Martín Vaquera

[29] Entornos de simulación gráfica para servicios OSGi. PFC de Jorge Herrera Rodríguez

APÉNDICE I: MANUAL DEL USO

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

APÉNDICE I.1: MANUAL DEL ADMINISTRADOR

En este apéndice se presenta cómo instalar los diversos componentes para la utilización de la aplicación.

APÉNDICE I.1: INSTALACIÓN Y EJECUCIÓN DE APACHE-FELIX

Para la ejecución de de nuestros bundles, es decir, de la aplicación, al estar funcionando sobre OSGi, utilizaremos la plataforma de arranque Felix que cumple con la especificación.

Felix, es totalmente gratuito y puede descargarse desde la página web del proyecto Apache en la siguiente URL:

http://felix.apache.org/site/downloads.cgi

Una vez en el link superior, elegiremos el .zip que aparece bajo el titulo “Felix Framework Distribution”.

Felix no necesita instalación, en el fichero descargado se incluyen todas las clases y paquetes necesarios para poder ejecutarlo directamente desde la consola cmd. Hay que navegar hasta situarse en la ruta donde hemos descomprimido el.zip, y ejecutar la sentencia:

Java –jar bin/felix.jar [18]

Permite trabajar por la línea de comandos de manera sencilla. En primer lugar nos ofrecerá un mensaje de bienvenida. A continuación, nos preguntará por un nombre para el perfil que queremos cargar, figura 33, ya que permite trabajar con diferentes perfiles que almacenan un registro de estado de los bundles. Este paso es configurable, y podemos indicarle que inicie siempre con un determinado nombre de perfil para no tener que indicárselo cada vez que ejecutamos el .jar. Para ello, en el fichero descargado, en la carpeta “conf”, aparece un fichero de configuración llamado “config.properties”. Añadiendo la línea felix.cache.profile=test, siempre se nos iniciará directamente con el perfil, en este caso, “test”. Esta configuración dependerá de nuestras necesidades. En este caso, cada vez que ejecutemos Felix lo haremos con un perfil diferente, ya que nos interesa para comprobar la funcionalidad de la aplicación.

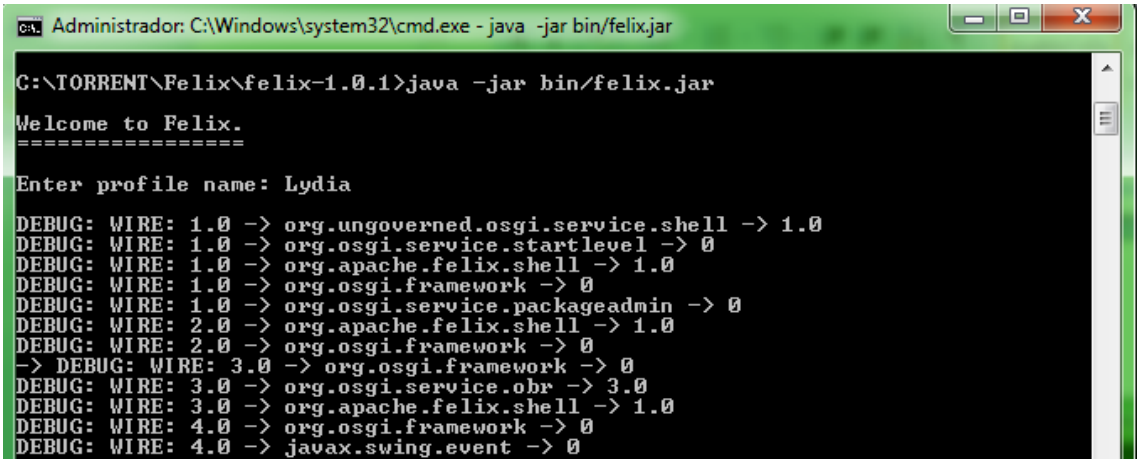

Figura 33: Pantalla de instalación de Felix

Si utilizamos el comando “help”, nos aparecerá una lista de comandos que podremos utilizar, figura 34.

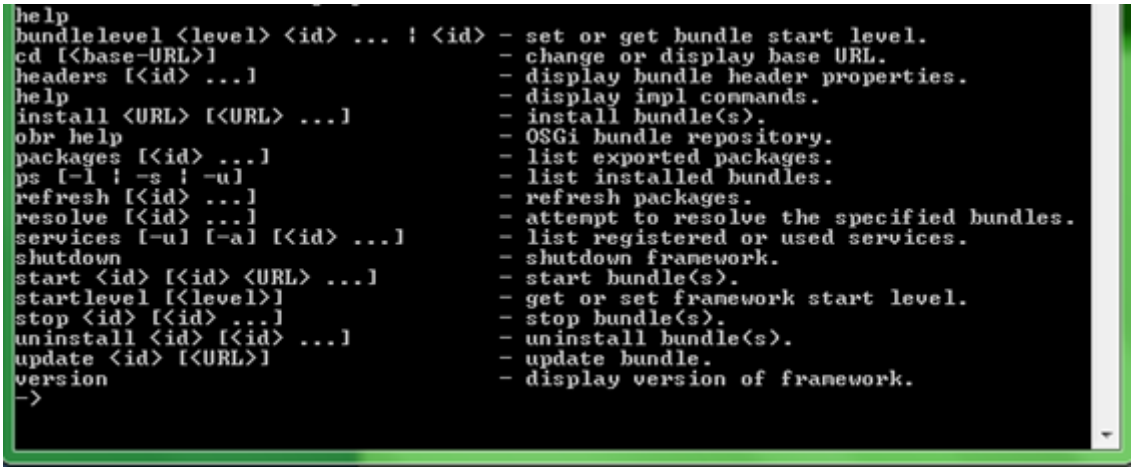

Figura 34: Pantalla del comando de ayuda

Los comandos que nos van a ser útiles son los siguientes:

ps

Muestra una lista de todos los *bundles* instalados, acompañados de un identificador y del estado actual en que se encuentra cada *bundle(Actived, Resolved…)*

install <ruta>

Instala el *bundle* indicado. Para que un *bundle* pueda ser instalado es necesario que se comprima en .jar, con su MANIFEST y su clase Activator.

unistall <id>

Desinstala el *bundle* indicado en el campo id..

start <id>

Arranca el *bundle* indicado en el campo id

stop <id>

Detiene el *bundle*

update <id><ruta>

Actualiza el *bundle* <id> con el indicado en <ruta>

APÉNDICE I.2: INSTALACIÓN DE LOS BUNDLES

--- --- --- --- --- --- --- ---

Instalado Felix, y una vez vistos los comandos necesarios para manejarnos, instalaremos los bundles necesarios de la aplicación para poder empezar a trabajar con ella.

Los nombres de los archivos .jar de cada *bundle* son los siguientes:

Tracker.jar: *Bundle* que contiene la ejecución del servidor para poder realizar la transferencia de archivos.

Torrent.jar: *Bundle* que contiene la implementación de los servicios upload y download, y las clases necesarias del protocolo BitTorrent.

Client.jar: *Bundle* que usa los servicios del Torrent.jar e implementa los comandos **[19]** para que interactúe el usuario.

Conocidos los nombres de los archivos, se procederá a su instalación. El comando install nos lo permite, y para ello tenemos que indicarle la ruta donde se encuentra el archivo. Se recomienda guardar los .jar en la carpeta C:/TORRENT/Felix/felix-1.0.1/bundle y de esta manera los comandos sirven para el sistema de ficheros del presente proyecto. Esto es porque al entrar en Felix, únicamente le indicaremos, en el caso de la instalación la carpeta de bundle/ como se ve a continuación:

install file: bundle/Tracker.jar

install file: bundle/Torrent.jar

install file: bundle/Client.jar

Introducidos estos comandos, ya tendremos los bundles instalados, como se observa en la figura 35:

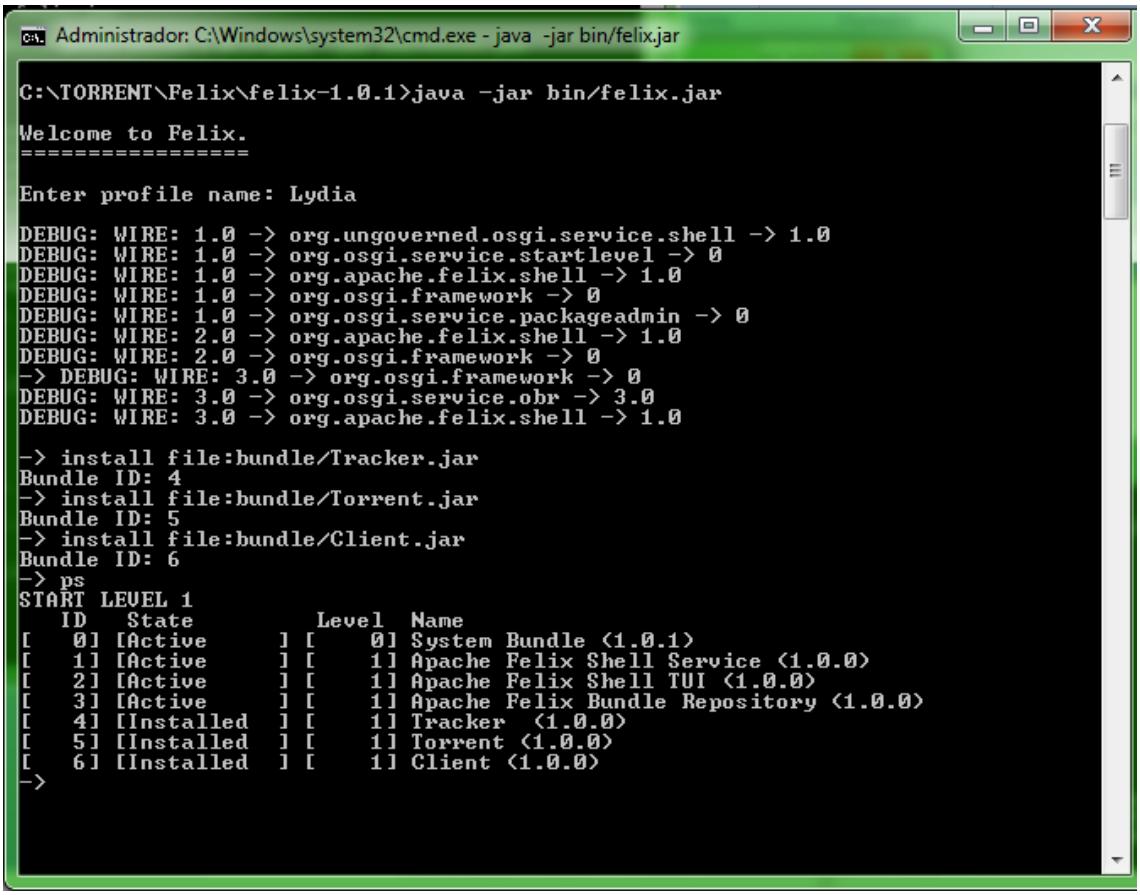

Figura 35: Pantalla Instalación de los bundles

Instalados los bundles, el siguiente paso a seguir es la ejecución del Tracker.jar, se introducirá start id <Tracker.jar>. El estado pasará a Active y el servidor estará preparado para escuchar y transferir los archivos que se le indiquen. El mensaje que nos aparecerá para saber que está funcionando correctamente es el siguiente: “Tracker started! Listening on port: 8081”

El siguiente paso, es ejecutar el servicio Torrent, para que pueda ser usado por los clientes.

Figura 36: Pantalla Ejecución Torrent.jar

Al llamar al start del bundle, nos indicará que los servicios han quedado registrados y por tanto, ya estarán listos para ser usados, y pasará de estado Installed a Active.

El último paso, es ejecutar el Client.jar, ya que es el que permitirá al usuario realizar las funciones de la aplicación y usar los servicios. Los comandos quedarán registrados, y se nos indicará por pantalla, como se ve en la figura 37.

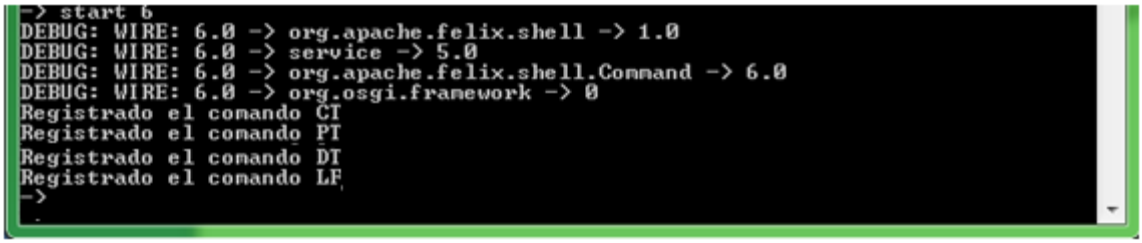

Figura 37: Pantalla Ejecución Client.jar

Activos los tres *bundles*, la aplicación queda disponible para el usuario. En el apéndice siguiente se explica la interacción.

APÉNDICE I.2: MANUAL DEL USUARIO

--- --- --- --- --- --- --- ---

En primer lugar el usuario deberá crear en su PC, en el directorio C una carpeta llamada TORRENT, y dentro de esta una carpeta UPLOAD, donde se guardarán los archivos que desea subir y los torrents que se creen correspondientes a estos archivos. A lo hora de crear el torrent es aconsejable crearlo con el mismo nombre del archivo para saber a quién está referenciado. También creará una carpeta DOWNLOAD que será donde se descarguen los archivos. Además para una mejor organización podremos crear dentro de DOWNLOAD otra carpeta cuando ejecutemos el comando DT.

C:/TORRENT/UPLOAD C:/TORRENT/DOWNLOAD

Los comandos de la aplicación del cliente han sido registrados por lo que el usuario lo primero que debe hacer es mirar la ayuda con el comando help donde se explica los que hace cada comando, figura 38.

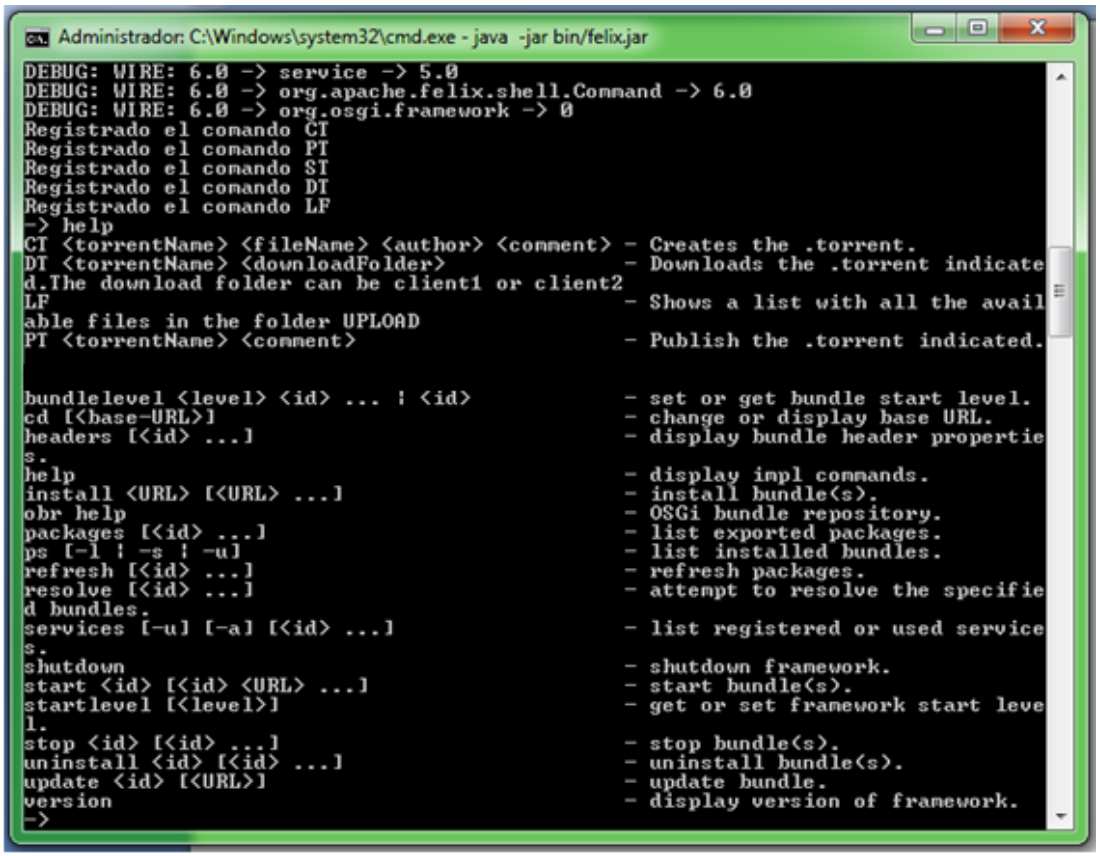

Figura 38: Pantalla help: comandos disponibles

Con el uso de los 3 primeros comandos que aparecen en la Figura 38, un cliente ya puede manejar la aplicación.

Para que el usuario comprenda mejor el funcionamiento, se va a presentar los pasos que realizaría un cliente que quiera subir un archivo, y otro usuario que se lo quiera descargar. Una vez visto este ejemplo, que es el más sencillo, se mostrará un ejemplo más complejo, y por último se indicarán las diferentes variantes de ejecución de los comandos para el correcto funcionamiento del flujo de transferencia.

Ejemplo básico:

Usuario1 decide subir una foto llamada ***imagen.jpg***, para que esté disponible para otro usuario que se la quiera descargar. **Usuario2**, mira la lista de archivos disponibles y al ver el torrent de la ***imagen.jpg***, decide descargársela.

Pasos a seguir:

Usuario1:

Paso 1: crear el torrent correspondiente a la ***imagen.jpg***. Ejecuta el comando CT con la información que se pide (ver figura 34).

CT *imagen.torrent imagen.jpg usuario1 foto*

Paso 2: publicar el torrent creado en el tracker y comnpartirlo para que pueda ser descargado. Ejecuta el comando PT con la información que se pide (ver figura 34).

PT *imagen.torrent foto*

Usuario2:

Paso 1: Primero tendrá que ver los torrents disponibles en el directorio. Ejecuta el comando **LF.** Aparecerá una lista con la *imagen.torrent.*

Paso 2: Decide que quiere descargarse el archivo **imagen.jpg**, asi que para ello ejecuta el comando DT con la información necesaria (ver figura 34).

DT *imagen.torrent carpeta1*

El **Usuario2**, ya tendrá disponible para él la foto ***imagen.jpg*** que había subido el **Usuario1**. El directorio dónde se encuentran los archivos disponibles es C:/TORRENT/UPLOAD. Y al usuario2 descargaría en el directorio C: /TORRENT/DOWNLOAD/carpeta1/.

Ejemplo con más interacciones:

Vamos a suponer, que un **Usuario1** quiere publicar dos archivos (***video.avi, canción.mp3***). A su vez, un **Usuario2** decide crear un torrent del archivo ***texto.txt***, es decir lo anuncia en el tracker pero no lo publica. El **Usuario1**, observa que está disponible el *texto.torrent* y decide publicarlo y compartirlo para poder descargárselo, ya que al no estar publicado ni compartido si intenta descargárselo directamente el mensaje de error que recibiría sería:

"The requested torrent is not listed on this Tracker"

El **Usuario2**, quiere descargarse el ***video.avi*** y la ***cancion.mp3***, sin embargo el **Usuario1** solo ha compartido el *video.torrent*, por lo que el **Usuario2**, podrá descargarse directamente el ***video.avi***, y para descargarse la ***cancion.mp3***, tendrá que compartirla y una vez compartida, descargársela.

Pasos a seguir:

Usuario1:

CT *video.torrent video.avi usuario1 video*

CT *cancion.torrent cancion.mp3 usuario1 cancion*

PT *video.torrent foto*

PT *cancion.torrent foto*

(El orden de introducción de los comandos puede variar. Podría haber creado y publicado el video y luego la canción y viceversa).

Usuario2:

CT *texto.torrent texto.txt usuario2 texto*

Usuario1:

LF

PT *texto.torrent texto*

DT *texto.torrent textos/*

(En este punto, el **Usuario1** ya se ha descargado el ***texto.txt*** del **Usuario2** y el ***video.avi*** y la canción ***song.mp3*** están disponibles para ser descargados)

Usuario2:

LF

DT *video.torrent videos/*

DT *cancion.torrent canciones/*

(En este punto, el ejemplo que da completado. Al final el **Usuario1** en su directorio *C:/TORRENT/DOWNLOAD/textos* se ha descargado el ***texto.txt***, y el **Usuario2** en su directorio *C:/TORRENT/DOWNLOAD/videos* se ha descargado el ***video.avi*** y en el directorio *C:/TORRENT/DOWNLOAD/canciones* se ha descargado la canción ***cancion.mp3***).

Variantes de ejecución:

Visto el ejemplo anterior, anotar, que la creación, publicación/compartición y descarga puede ser independiente, y puede llevarse a cabo cada paso por diferentes clientes siempre y cuando se esté trabajando con el mismo tracker. No obstante, anotar, que los pasos de publicación y compartición, es recomendable que se hagan seguidos ya que si un archivo publicado va ser descargado y este no ha sido compartido, nos dará error, por ello la implementación interna está configurada para que al publicar se esté compartiendo directamente.

APÉNDICE II: DESCRIPCIÓN DEL CD

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

El CD contiene las siguientes carpetas:

TORRENT: carpeta que contiene toda la información de los *bundles*. Dentro de ella hay una carpeta por cada *bundle* de la aplicación, y dentro de cada una de estas carpetas aparecen las siguientes:

- ***bin***: contiene los *.class*, es decir el código compilado
- ***lib***: contiene el archivo *.jar* que representa el *bundle*
- ***src***: contiene los ficheros de código
- Además están incluidos en la carpeta de cada *bundle*, el build.xml para la creación del .jar y el MANIFEST de cada *bundle*.

Dentro de la carpeta TORRENT aparecen también las siguientes:

- ***Memoria***: contiene la memoria del proyecto Servicio de BitTorrent para plataformas de servicios OSGi.
- ***Felix***: copia del programa Felix, versión felix_1.0.1. esta carpeta a su vez contiene:
  - bin: ficheros felix.jar que almacena las librerías de OSGi.
  - bundle:carpeta donde se almacenan los bundles de la aplicación generados por eclipse.
  - conf: contiene el archivo de configuración del frmawork de OSGi, config.properties
  - cache: contiene los perfiles de configuración que genere el usuario de la aplicación.
- ***UPLOAD*** carpeta donde cargarán los archivos los usuarios
- ***DOWNLOAD*** carpeta donde los usuarios obtendrás las descargas de sus archivos.